2003

Parallel stochastic context-free grammar training for tRNA secondary structure prediction.

Kaiyuan. Shi
University of Windsor

Parallel Stochastic Context-Free Grammar Training

For tRNA Secondary Structure Prediction

by

Kaiyuan Shi

A Thesis
Submitted to the Faculty of Graduate Studies and Research
through the School of Computer Science
in Partial Fulfillment of the Requirement for the Degree of Master of Science
at the University of Windsor

Windsor, Ontario, Canada
Summer, 2003

Your file Votre référence
ISBN: 0-612-84555-9
Our file Notre référence
ISBN: 0-612-84555-9

ABSTRACT

In this thesis, parallel computing methods are used to construct a stochastic context-free grammar for analyzing the secondary structure of tRNA molecules.

Stochastic context-free grammars are basically probabilistic languages that parse a sequence/molecule and output its probability of whether or not it belongs to the sequence family modeled by the grammar and at the same time predict the molecule's secondary structure. Stochastic context-free grammars can be converted from corresponding context-free grammars, by assigning probabilities to the grammar's production rules.

The use of stochastic context-free grammars to analyze the secondary structure of RNA molecules was limited due to time and space complexity.

In order to overcome such problems, we apply the parallel computing method to train a stochastic context-free grammar and use it to predict the secondary structures of tRNA molecules. The test results demonstrate that our parallel method is efficient and produces results that are at least better than current sequential methods.

ACKNOWLEDGEMENTS

I would like to express my sincere thanks and gratitude to Dr. Alioune Ngom for his invaluable guidance, instruction and help through the course of this thesis without which this thesis would not be accomplished.

Gratitude is also expressed to Dr. Jianguo Lu, Dr. Adnan Ali, Dr. Richard A Frost, and Dr. Robert D Kent; all offered much help and useful suggestions during this thesis project.

I also want to thank my graduate fellows at the Department of Computer Science for the discussion and help in my studies.

Great appreciation is expressed to my parents, my wife and my baby daughter for their continued support and encouragement.

DEDICATION

To My Family

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

A	Adenine
C	Cytosine
G	Guanine
T	Thymine
U	Uracil
DNA	Deoxyribonucleic acid
RNA	Ribonucleic acid
mRNA	Messager RNA
rRNA	Ribosomal RNA
tRNA	Transfer RNA
CM	Covariance Model
CYK	the Cocke-Younger-Kasami algorithm
EM	Expectation-Maximization
HMM	Hidden Markov model
MPI	Message Passing Interface
SCFG	Stochastic Context-Free Grammar

1 INTRODUCTION

1.1 The RNA World

RNA is the only biological polymer that serves as both a catalyst (like proteins) and as information storage (like DNA). For this reason, it was assumed that RNA was the basis of life early in evolution. This led to the theory that RNA based organisms pre-existed before DNA and protein based organisms which currently crowd Earth (Woese1985).

They established what is now called the RNA-World hypothesis, a world in which RNA catalyzed virtually all the reactions necessary for a precursor of life's last common ancestor to survive and replicate. The motivation for favoring RNA over DNA as the medium for coding genetic information is that RNA is considerably easier to synthesize than DNA.

The discovery by Cech (Cech1986) that RNA is sufficiently chemically active to catalyze the splicing and cleavage reaction of other strands provides considerable weight to the RNA-World theory. Cech's discovery meant that not only could RNA have been the sole form of genetic storage in an RNA-World, but it could also have performed many of the enzymatic roles that the proteins perform. An interesting consequence of this theory is that many of the metabolically active RNAs may in fact be relics or molecular fossils of this RNA-World.

There are three major types of RNA: messenger RNA (mRNA), transfer RNA (tRNA) and ribosomal RNA (rRNA) found inside the cells. A number of other types of RNA are present in smaller quantities as well, including small nuclear RNA (snRNA), small nucleolar RNA (snoRNA) and the signal recognition particle RNA (srpRNA).

Excluding the RNA-World theory, RNA serves a multitude of roles in living cells. These include: serving as a temporary copy of genes that is used as a template for protein

synthesis (mRNA), functioning as adaptor molecules that decode the genetic code (tRNA) and catalyzing the synthesis of proteins (rRNA). There is much evidence implicating RNA structure in biological regulation and catalysis.

With no uncertainty, RNA molecules play an important role in the living world. Thus, how to determine their structure and deduce their functionalities has been an important problem in biology.

1.2 RNA Structures

RNA molecules are polymers of nucleotides that are chains of ribonucleotide monophosphates covalently joined together by phosphodiester linkages. RNA molecules are a single chain, called strand consisting of same basic unit, which contains a sugar molecule, 2'-ribose, attached to a phosphate residue. The sugar molecule consists of carbon atoms, which are labeled 1' through 5'.

RNA molecules have an orientation starting at the 5' side and ending at 3' side. We use 5'→3' to represent the direction of a single stranded RNA sequence. Figure1.1 depicts a segment of the sugar-phosphate backbone of RNA molecule.

RNA molecules can form primary (linear, 1-dimentional), secondary (planar, 2-dimentional), tertiary (3-dimentional) and quaternary structures. The secondary structures are formed mainly due to the complimentary base pairings. The tertiary structures, based on secondary structures, are formed due to the interactions among some of the remote non-complimentary base pairings. The quaternary structures are formed due to the interactions with other binding proteins.

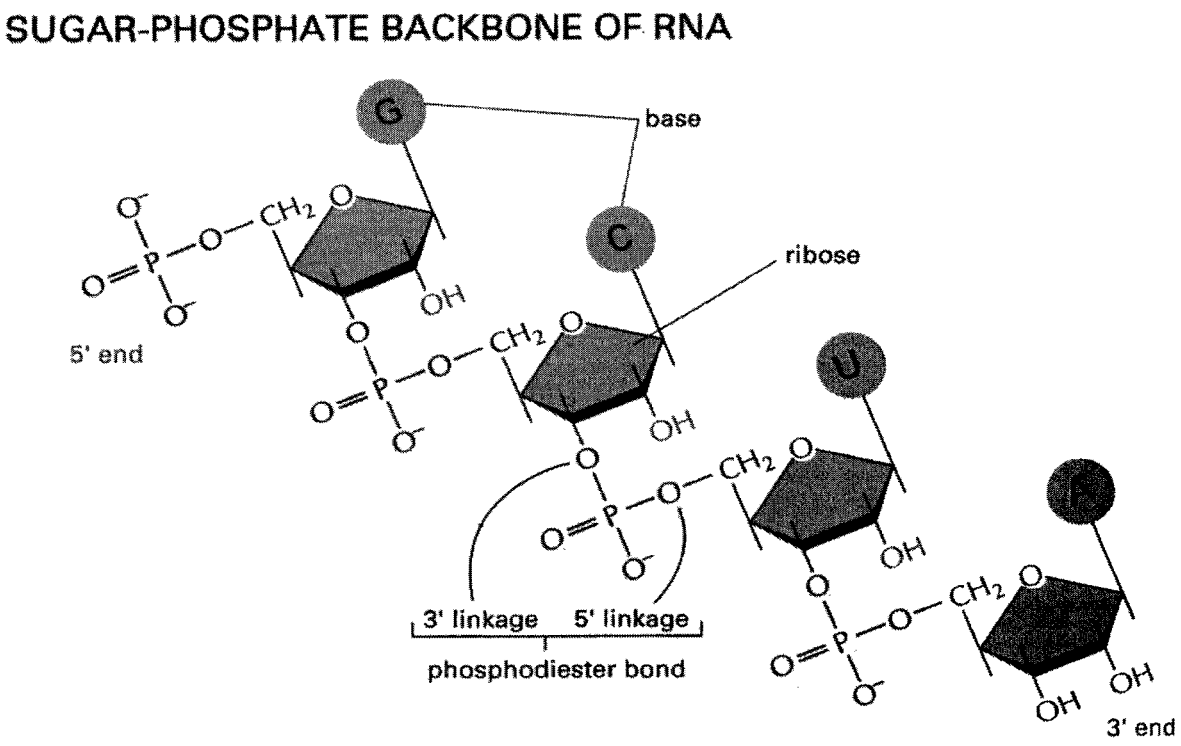

Figure1.1: The sugar-phosphate backbone of RNA molecule (Adopted from Alberts1994).

There are four major nucleotide bases found in RNA molecules, including Adenine (A), Guanine (G), Cytosine (C), and Uracil (U). Figure1.2 depicts the structures of the bases found in RNA.

Some RNA molecules also contain modified bases. Figure1.3 depicts the structures of the modified bases found in RNA.

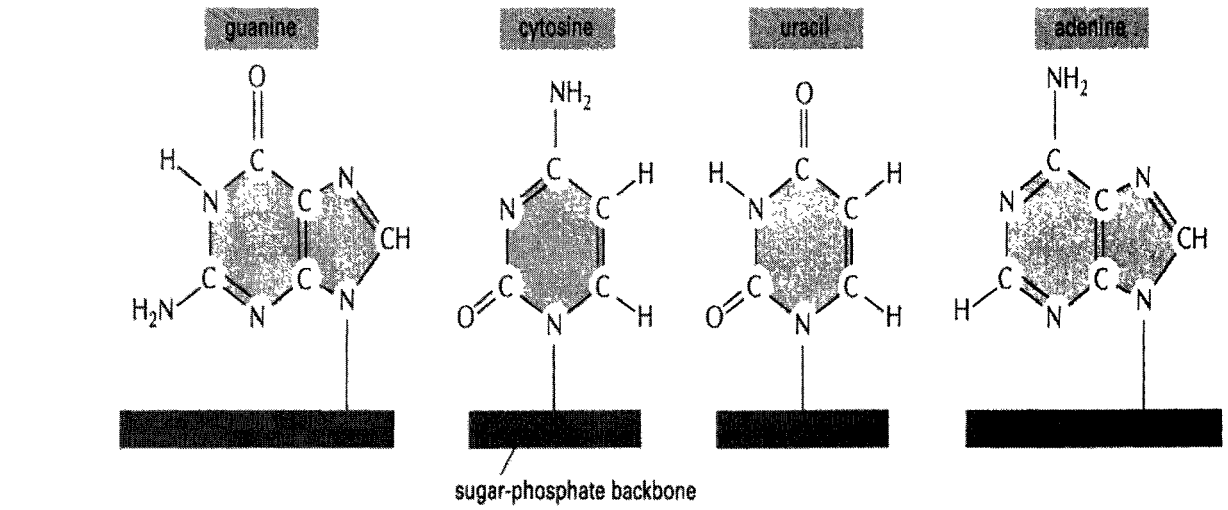

Figure1.2: The structures of the major nucleotide bases found in RNA (Adopted from Alberts1994)

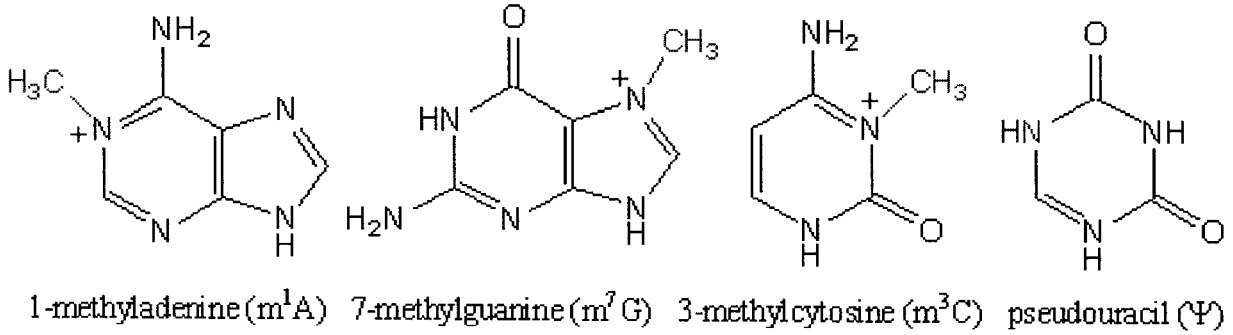

Figure 1.3: Examples of modified nucleotide bases found in RNA (Modified from Voet1999)

1.2.1 Primary Structure

The phosphodiester linkage of monomer units into a polymeric molecule is called the primary structure or linear structure. The phosphodiester linkages have a direction (called polarity; the convention is 5' to 3' going from left to right). Figure 1.4 depicts a primary structural formation of an RNA segment.

The arrangement (or order) of specific nucleotides along the chain is called the sequence. The sequence contains the genetic information. A sequence can be written simply either as: CAGC…GACG or 5'- CAGC…GACG -3'.

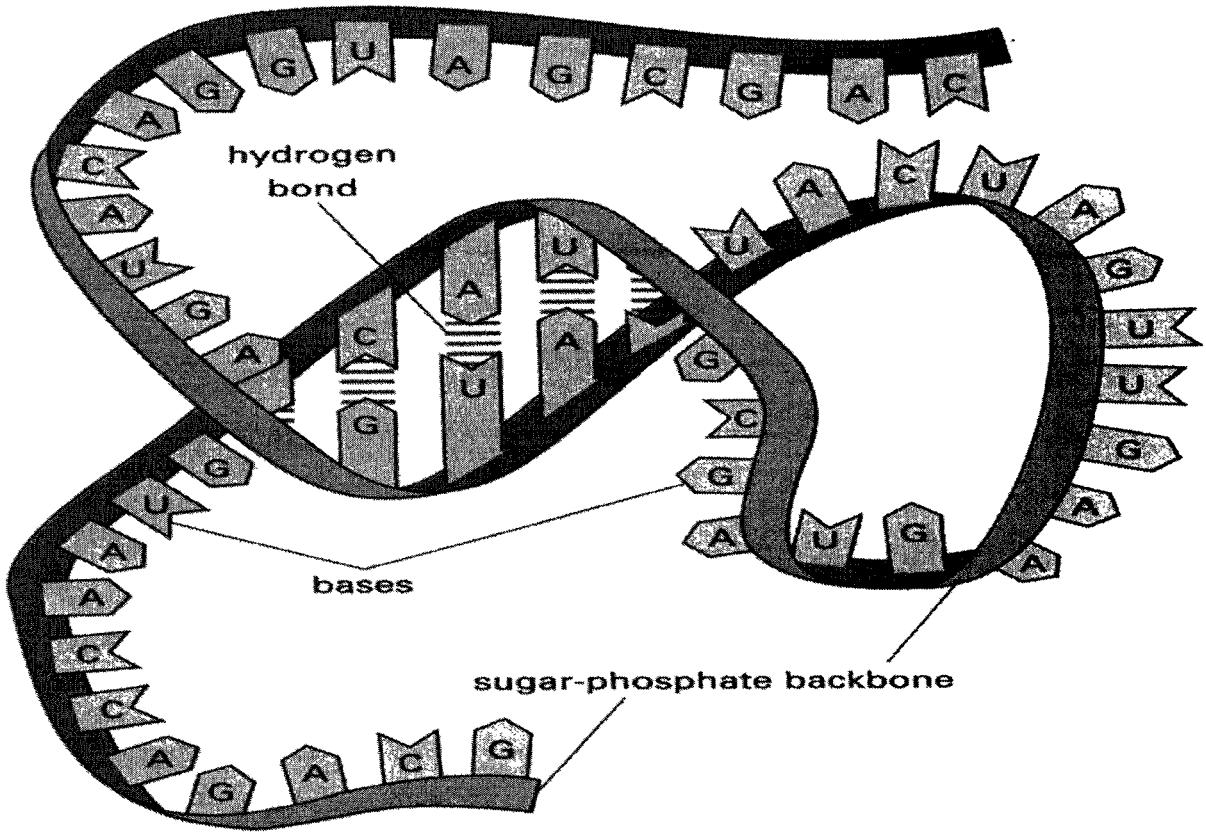

Figure 1.4: An example of the primary structure formation of a RNA segment

(Adopted from Alberts1994).

1.2.2 Secondary Structure

In RNA molecules, the nucleotides A, C, G and U interact in specific ways to form characteristic secondary structure motifs such as helices, loops and bulges. The canonical base pairing are "A" hydrogen bonds with "U" and "G" hydrogen bonds with "C".

Sometimes "G" bonds with "U". Figure 1.5 depicts the interaction between the nucleotides G and C, A and T(in case of RNA, the base "T" should be replaced by U).

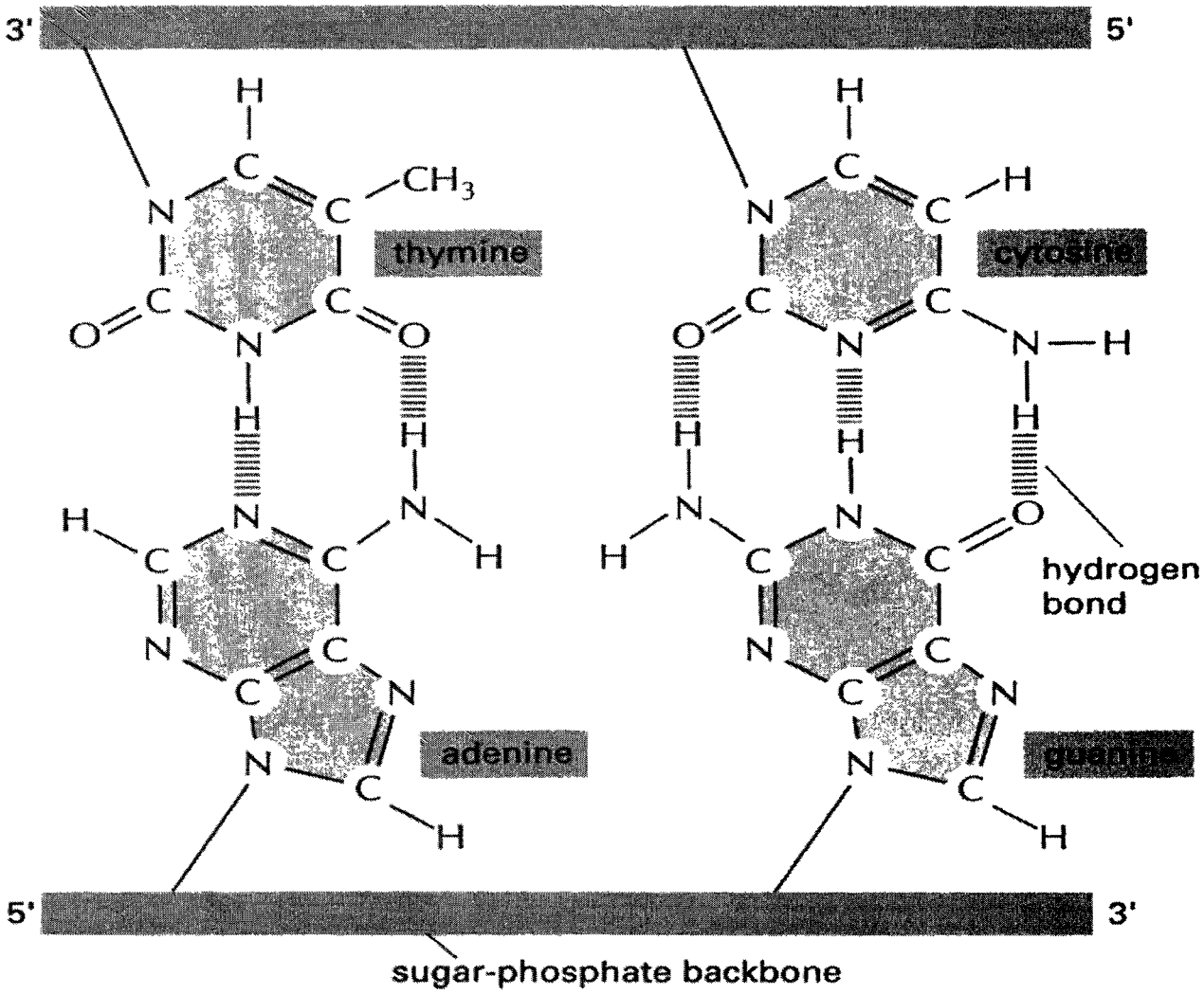

Figure 1.5: Specific hydrogen bonding between G and C and between A and T (U for RNA molecules), Adopted from Alberts1994.

Base pairings between nucleotides of the primary structure result in various elements of secondary structure known as loops, stems (helices), and hairpins. They are defined as:

1. Contiguous stacked base pairs (called stems).

2. Single stranded subsequences bounded by base pairs (called loops). Simple substructures consisting of a simple stem and loop are called stem loops or hairpins.

3. Single stranded bases occurring within a stem (called bulge loops).

4. Interior loop, if there are single stranded bases on both sides of the stem.

5. Multi-branched loops, from which three or more stems radiate.

Figure 1.6 shows a hypothetical example of the fundamental elements of RNA secondary structure.

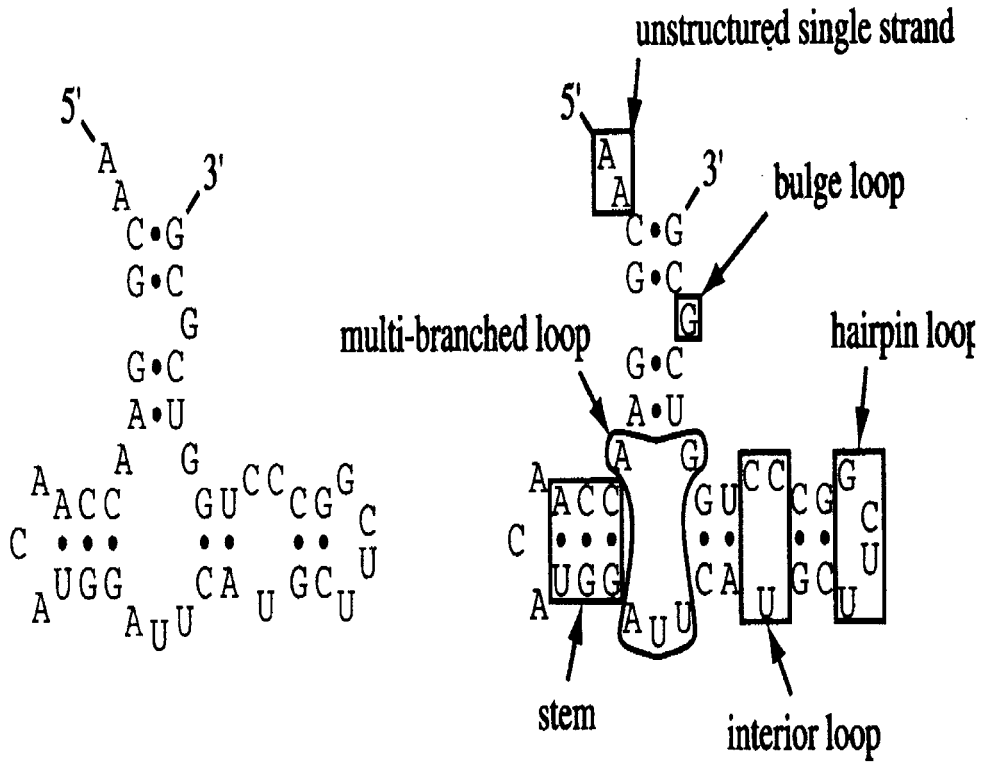

Figure 1.6: A hypothetical example of the fundamental elements of RNA secondary structure (Adopted from Durbin1998)

RNA Secondary Structure is frequently represented as a planar graph satisfying a few basic criteria.

Figure 1.7 depicts a secondary structure of a tRNA molecule which includes several secondary structure elements that characterized the tRNA secondary structure: AA-Stem, the amino acid acceptor stem; D-Arm, the dihydrouridine hairpin; AC-Arm, the anticodon hairpin; V-Loop, the variable loop; T-Arm, the T ΨC hairpin.

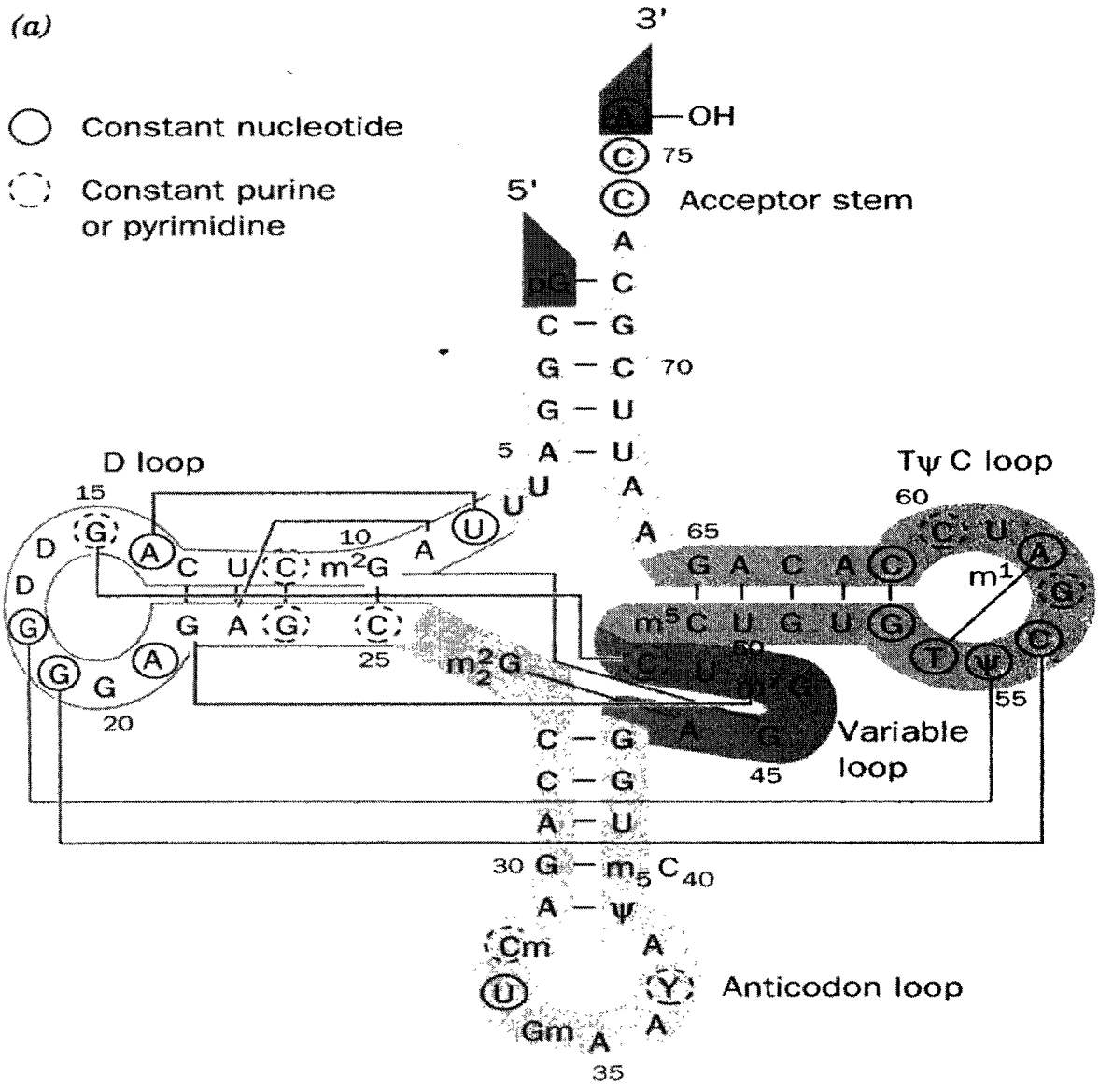

Figure 1.7: An example of tRNA cleaver secondary structure (Adopted from Voet1995)

1.2.3 Tertiary and Quaternary Structure

Further folding and hydrogen-bonding interactions between remote regions orient secondary structure elements with respect to each other to form the tertiary structure and thus determine the functional system. Higher-order interactions with other proteins or nucleic acids may also occur, called quaternary structure in this case.

In tertiary structure, many of these interactions involve non-canonical base pairing and/or interactions involving three or more bases. The base pairings, "G" pairing with "C" and "A" pairing with "U" are called canonical base pairings or Watson-Crick base pairings. Other base pairings, such as "G" pairing with "U" and "G" pairing with "A", are called non-canonical base pairings.

Base pairs between a loop and positions outside the enclosing stem are called pseudoknot, which can be regarded as one of the tertiary structure elements. Figure 1.8 illustrates an example of pseudoknot.

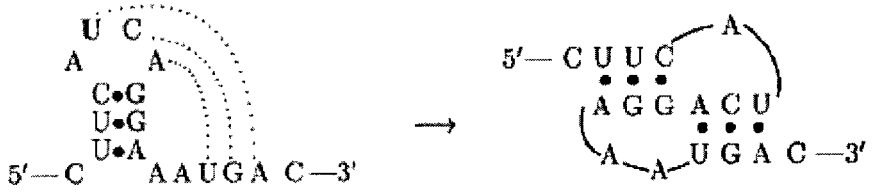

Figure 1.8: An example of Pseudoknots (Adopted from Rivas1999). Left: Base pairs between a loop and positions outside the enclosing stem; Right: another representation of the same pseudoknot.

Figure 1.9 depicts an L-shaped tertiary structure of $tRNA^{Phe}$ molecule.

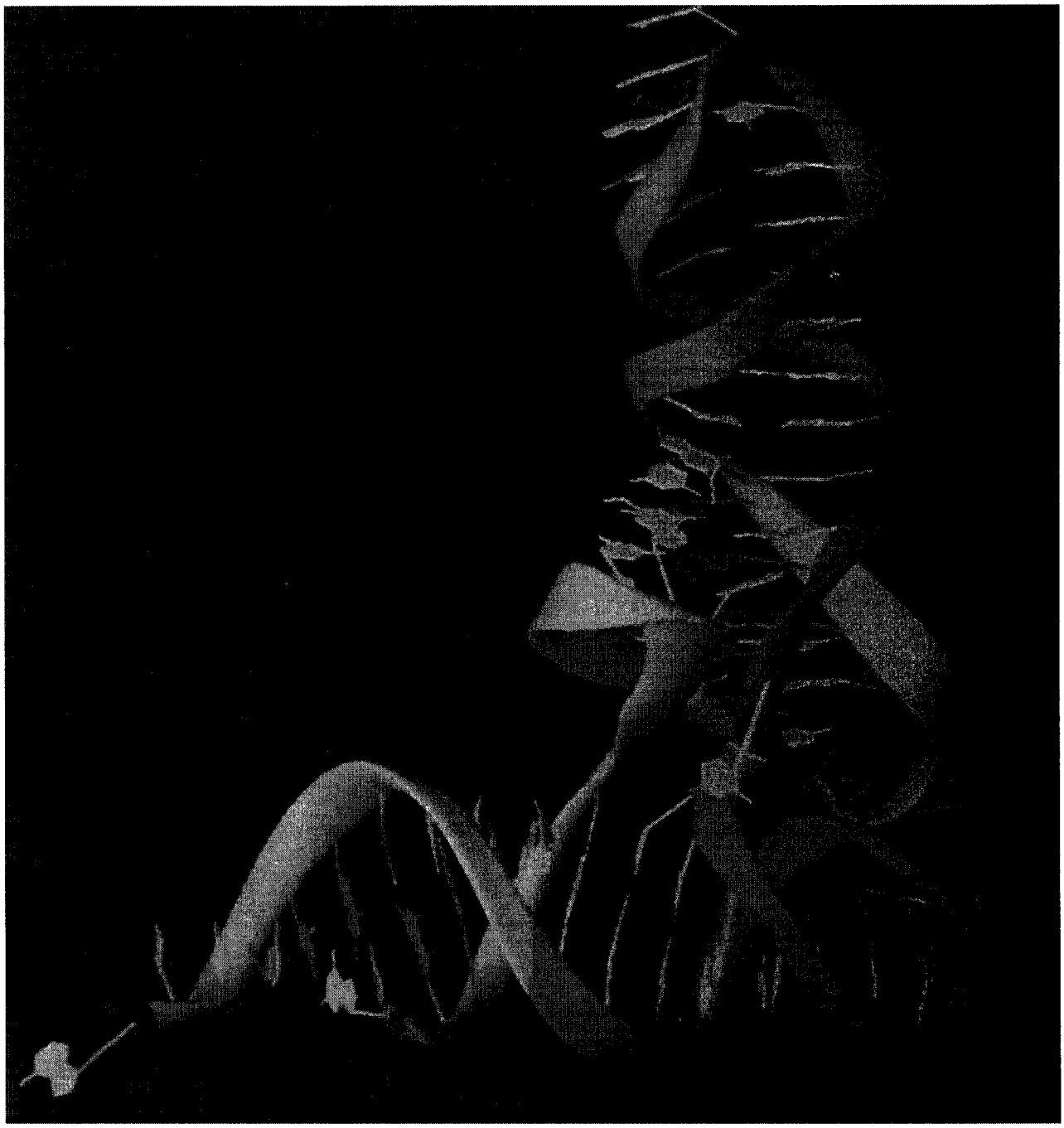

Figure 1.9: An example of tRNA tertiary structure (Adopted from Voet1995).

1.3 RNA Secondary Structure Prediction

The folding of the primary structure into tertiary structure can be decomposed into two steps:

1. The formation of the RNA secondary structures by the Watson-Crick base-pairings (G-C and A-U), the weaker G-U pairs and more rarely the G-A base pairs; such base pairs constitute the major part of RNA secondary structure.
2. Folding of the planar secondary structure into a three-dimensional tertiary structure in the presence of divalent metal ions such as Mg^{2+}.

The intramolecular forces stabilizing the secondary structures are much stronger than those driving the arrangement of the secondary structure elements in 3D space.

The dominance of secondary structure formation over tertiary structure formation is supported by the well-documented conservation of secondary structure in evolution.

On the other hand, RNA functions are decided by their higher order structures. However, different primary structures may have same higher order structure, such as the secondary structure, due to base mutations. Therefore we cannot determine the function of an RNA molecule from its primary structure only.

Thus a good place to study the function of RNA molecules is at the level of secondary structure.

The remaining chapters are organized as following: in chapter 2, we review the major current methods/techniques proposed to solve the RNA secondary structure prediction problem. In chapter 3, we describe our methods to predict tRNA secondary structure. In chapter 4, we describe the implementations and our test results. In chapter 5, we summarize the future work. In chapter 6, we summarize our works.

2 EXISTING METHODS FOR RNA SECONDARY STRUCTURE PREDICTION

There are three dominant methods for predicting the secondary structure of RNA molecules, including the comparative sequence analysis, the free energy minimization method and linguistic method (Stochastic context free grammar).

The comparative sequence methods have been generally more robust for large RNA molecules (James1989, Noller1981A, Noller1981B, Pace1989, Shapiro1990, and Stephan2000). However, using phylogenetic information to predict the secondary structure of RNA relies upon a sequence alignment and knowledge of the consensus secondary structure of homologous RNAs. This alignment step often requires labor-intensive manual intervention. Often homologous RNA sequences (or structures) are unavailable, thus making the comparative sequence analysis method unrealistic for large families of RNA.

The free energy minimization method relies on thermodynamic parameters and dynamic programming to determine near minimum free energy secondary structures (Antao1992, Jaeger1990, Le1991, Nussinov1980, Papanicolaou1984, Tinoco1971, Turner1988, Zuker1981, Zuker1984, Zuker1989, and Zuker2000). The driving force behind secondary structure formation is the stacking of base pairs. The formation of an energetically favorable helical region also implies the formation of an energetically unfavorable loop region. This frustrated energetic leads to a vast combinatory of helices and loop arrangements.

Stochastic context free grammar (SCFG) can characterize RNA sequences through that it take into account the statistical identity of different sequence positions including pairwise interactions and thus capture the sequences' common primary and secondary structures

(Brown1995, Eddy1994, Eddy2002, Grate1994, Knudsen1999, Leslie1995, Sakakibara1994). SCFG can be used to determine the free energy of a molecule's secondary structure.

Covariance Models (CMs), implemented in the software package COVE by Eddy (Eddy1994), are probabilistic models and can be viewed as generalization of Hidden Markov model (HMM), which is a specialized CM with no bifurcation and pairwise states for describing covariations.

The difference between SCFGs and CMs is that SCFGs typically emit symbols on transitions while CMs emit symbols on state (Durbin1998).

CMs are equivalent to SCFGs in essence (Eddy1994, Sakakibara1994).

The following sections describe in detail the above methods.

2.1 Comparative Sequence Analysis

Comparative analyses of two or more RNA sequences have been used widely in detection and evaluation of biological similarities and evolutionary relationships and thus predicting the secondary structure of functional RNA molecules.

Comparative sequence analyses have been used to predict the structure of tRNAs, rRNAs, and a number of ribozymes. This form of analysis can be used to predict secondary structure pairings, and even some tertiary interactions. Many results were later verified when the structure of the molecules were obtained. For example the crystal structure, which was determined by X-ray diffraction, of yeast Phe-tRNA verified all predicted interactions (Kim1974). Several structural elements such as pseudoknots and non-canonical pairings have been proposed on the basis of comparative analysis, and were substantiated by high-resolution experimental methods.

The principle of comparative sequence analysis is based upon the assumption that the secondary structure of a functional RNA will be conserved better than the primary structure during the evolution of the RNA. This assumption is reasonable as it is very unlikely that a molecule will undergo a total change in structure during its evolution and still remains functional. The consensus secondary structure of a family of homologous RNA molecules can be deduced by comparing the individual structures of the sequences.

Figure 2.1 illustrates an example of base pair leading to a consensus structure prediction. The two boxed-positions in this example of a multiple alignment are covarying to maintain Watson-Crick complementarities.

Constraints upon secondary structures can be revealed by compensatory base changes, when a base changes in a double stranded region then often the complementary base will change during the evolutionary history of the RNA to compensate for the original mutation. If compensatory mutations did not occur this would result in loss of secondary structure and in eventual loss of functionality, which would be selected against. Several compensatory base changes can be generally considered as proof of a possible interaction. This approach can be applied to predict canonical base-pairings and even tertiary interactions.

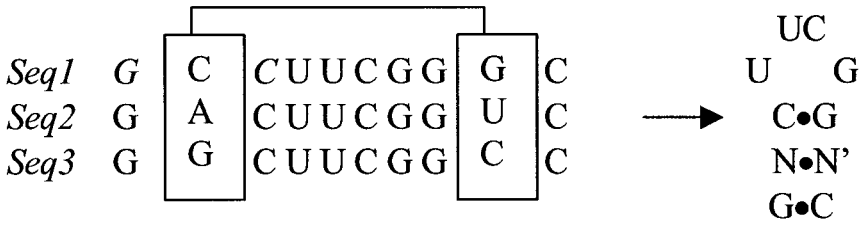

Figure 2.1: An example of a base pair leading to a consensus structure prediction

(Modified from Durbin1998)

The original procedure of Noller and Woese (Noller1981) detected compensatory base changes in putative helical elements. Positions that covaried were assumed to be base-paired. This procedure was subsequently formalized into an explicit computer algorithm (Waterman1986).

The algorithm of Sankoff (Sankoff1985), for simultaneously aligning and folding sequences, is generally impractical in terms of time and space for large numbers of long sequences. Given an alignment of homologous RNA sequences, heuristic methods have been proposed to predict a common secondary structure (Han1993).

However, there remains no reliable or automatic way of inferring an optimal consensus secondary structure even if the related sequences are already aligned. Because considerable manual intervention is still required to identify potential helices that maintain base complementarities. Automation and development of more rigorous comparative sequence analysis protocols are under continual development.

2.2 Free Energy Minimization Method

The free energy minimization method employs thermodynamics to compare the free energy changes predicted for formation of possible secondary structure, and relies on finding the structure with the lowest free energy, that is the most stable structure. Such energy minimization depends on thermodynamic parameters and computer algorithms to find the optimal free energy folding of an RNA molecule (Jaeger1990).

To obtain a common folding pattern for a set of related molecules, Zuker (Zuker1984) has suggested predicting a folding for each sequence separately using these algorithms and then searching for a common structure (Zuker1989). Limitations of this method are partially due to the uncertainty in the underlying energy model, and the technique may be overly sensitive to point mutations (single base mutations). Attempts were made to combine both phylogenetic and energetic approaches (Le1991).

Since RNA molecules comply with the laws of thermodynamics, it should theoretically be possible to deduce the structure of an RNA molecule from its sequence by locating the conformation with the lowest free energy. The advantage of this approach is that it does not require a multiple sequence alignment, which is often difficult to produce. In fact energy minimization techniques only require single sequences.

Experimental energy parameters are available for the contribution of individual loop as a function of its size, of the type of its delimiting base pairs, and partly of the sequence of the unpaired strains.

An alternative to using laboratory techniques to determine thermodynamic folding parameters is to use known biological structures and iteratively update the energy parameters until the correct secondary structure is predicted. Tinoco (Tinoco1971) was

among the first to experimentally estimate thermodynamic parameters for RNA secondary structure prediction.

However, it is not feasible to study the thermodynamics of every possible sequence, so a simplified model is necessary to estimate the properties of many sequences from data obtained using a limited number of sequences. The most popular thermodynamic model is the nearest neighbor model. Since hydrogen bonding and stacking are both short-range interactions, the stability is assumed to depend only on the identity of the adjacent pairs. This model has been experimentally validated as a reasonable approximation for many cases. Effects of loops and bulges are a little more difficult to estimate for free energy contribution. Originally these regions were assumed to be solely dependent upon the number of unpaired nucleotides they contained.

Zuker (Zuker1981) developed a recursive algorithm for obtaining an optimal folding for large RNA sequences that utilizes thermodynamic parameters. The advantage of this algorithm over other possible methods is speed, but to keep the algorithm computationally feasible several simplifications had to be made. For example tertiary interactions such as pseudoknot formation are ignored. Although a recent algorithm developed by Rivas and Eddy (Rivas1999) is now available that will predict pseudoknots for short RNA sequences (such as tRNA).

There are one or two significant problems associated with using minimum free energy (MFE) methods. Firstly the energy parameters, which the folding algorithm relies on, are inevitably imprecise because the parameters are induced manually by experiments. Hence, the true minimum free energy structure might be one that is suboptimal with respect to the parameters used. The same might hold because of unknown biological constraints that may

change relative energies, turning an otherwise suboptimal structure into the most favorable one. Also, under physiological conditions, RNA sequences may exist in alternative states whose energy difference is small.

The following sections discuss two major algorithms for free energy minimization.

2.2.1 The Nussinov Algorithm

The Nussinov algorithm (Nussinov1978, Nussinov1980) assigns a score to all possible secondary structures for a given primary sequence. The correct secondary structure gets the highest score, and can be eventually identified. The algorithm is recursive; it calculates the best structures for small subsequences, and works its way outwards to larger and larger sub-sequences.

According to the algorithm, there are only four possible ways of getting the best structure for a subsequence i, j from the best structures of smaller sub-sequences.

- Add unpaired position i onto the best structure for sub-sequence $i + 1, j$
- Add unpaired position j onto the best structure for sub-sequence $i, j - 1$
- Add i, j pair onto the best structure found for sub-sequence $i + 1, j - 1$
- Combine two optimal sub-structures i, k and $k + 1, j$.

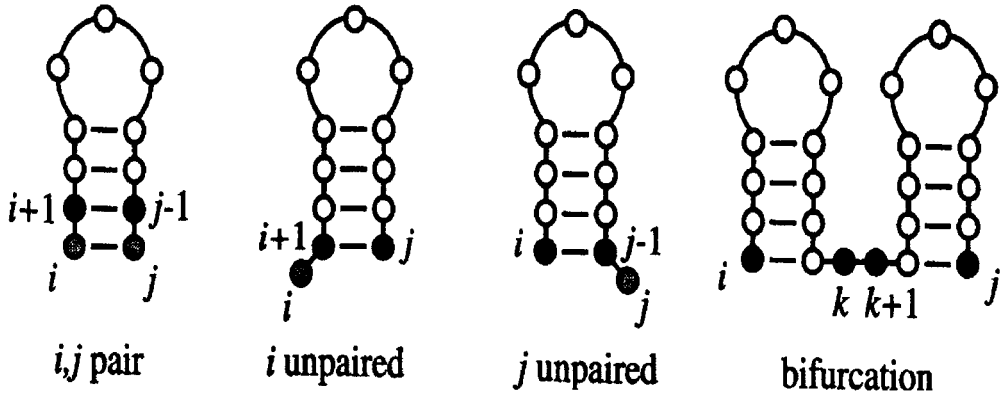

Figure 2.2: Four possible ways of getting the best structure for a subsequence i, j from the best structures of the smaller sub-sequences (Adopted from Durbin1998)

Figure 2.2 illustrates the four possible ways of getting the best structure for a subsequence i, j.

We use the term $\gamma(i, j)$ to denote the scores of the optimal structure that can be formed from subsequence $x_i, \ldots, x_j$. The Nussinov algorithm is summarized as follows:

Filling stage:

Initialisation:

$$\gamma(i, i-1) = 0 \qquad \text{for } i = 2 \text{ to } L;$$
$$\gamma(i, i) = 0 \qquad \text{for } i = 1 \text{ to } L.$$

Recursion: **starting with all subsequences of length 2, to length L:**

$$\gamma(i,j) = \max \begin{cases} \gamma(i+1, j), \\ \gamma(i, j-1), \\ \gamma(i+1, j-1) + \delta(i,j), \\ \max_{i<k<j} \left[\gamma(i,k) + \gamma(k+1, j)\right]. \end{cases}$$

Figure 2.3 illustrates an example of a Nussinov matrix fill in operation for the sequence GGGAAAUCC. Diagram (a) shows the initialized half-diagonal matrix; (b) shows the matrix after scores for subsequences of length two have been calculated; (c) shows an example of two different optimal substructures for the same subsequence; (d) shows the final matrix.

The value of $\gamma(1, L)$ is the number of base pairs in the maximally base-paired secondary structure(i.e. the best structure). In order to find one of the maximally base-paired structures (there are more than one maximally base-paired secondary structure), we recover a

traceback in the matrix beginning from $\gamma(1, L)$, along the highest scoring path and thus deduce the structure.

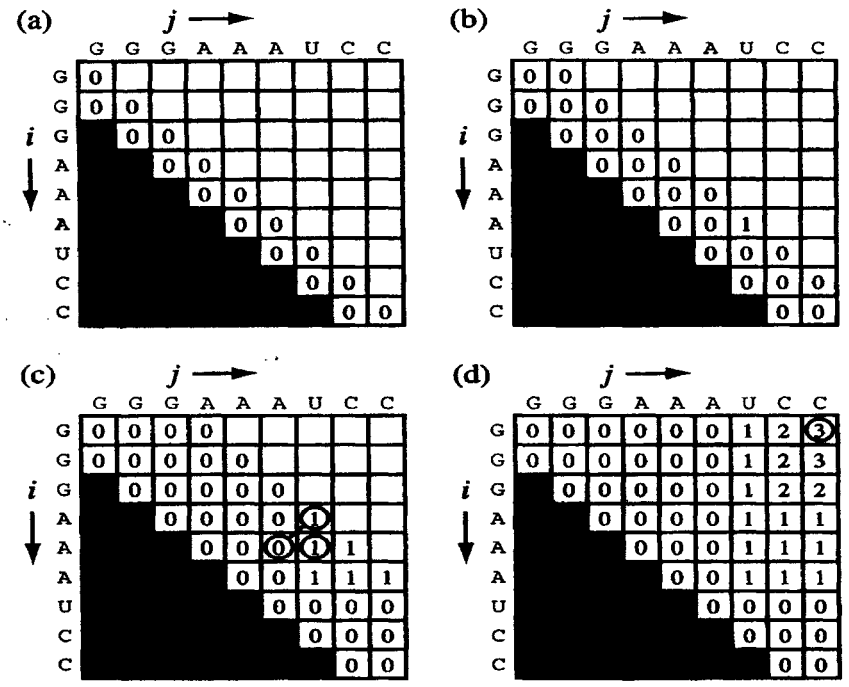

Figure 2.3: An example of a Nussinov matrix fill in operation (Adopted from Durbin1998)

The traceback algorithm is summarized as follows:

Initialisation: Push $(1, L)$ onto stack.
Recursion: Repeat until stack is empty:
- pop (i, j).
- if $i >= j$ continue;
  else if $\gamma(i+1, j) = \gamma(i, j)$ push $(i+1, j)$;
  else if $\gamma(i, j-1) = \gamma(i, j)$ push $(i, j-1)$;
  else if $\gamma(i+1, j-1) + \delta_{i,j} = \gamma(i, j)$:
    - record i, j base pair.
    - push $(i+1, j-1)$.
  else for $k = i+1$ to $j-1$: if $\gamma(i,k) + \gamma(k+1, j) = \gamma(i, j)$:
    - push $(k+1, j)$.
    - push (i, k).

Figure 2.4 shows an example of traceback stage for the filled matrix from Figure 2.3. The left part of the figure shows an optimal traceback path indicated with circles. The right part of the figure shows an optimal structure corresponding to the path.

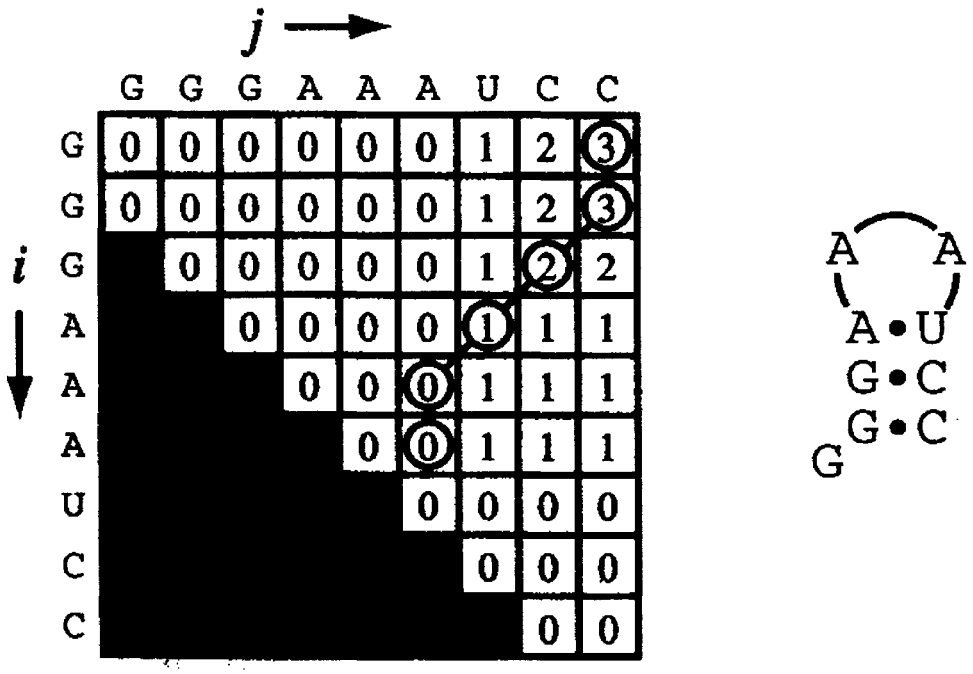

Figure 2.4: An example of the traceback for Nussinov algorithm (Adopted from Durbin1998)

2.2.2 The Zuker Algorithm

Zuker's algorithm (Zuker1981, Zuker1984) is based on rules of structure determination by free-energy calculations proposed by Waterman (Waterman1978), who noted that newer approaches of structure determination revolved around the fact that the most probable structure should be the one with the lowest energy.

Waterman's free-energy calculations:

Let $h(i, j)$ be the minimum free-energy of a hairpin secondary structure on subsequence $a_i a_{i+1} \ldots a_j$, $i < j$, where a_i and a_j form a base pair and there is a single ended loop. If a_i and a_j cannot form a basepair, $h(i, j) = +\infty$.

The free energy functions are of the form:

- $\alpha(a_i, a_j)$ = free energy of an base pair(a_i, a_j);
- $\xi(k)$ = destabilization free-energy of an end-loop of k bases;
- η = stacking energy of adjacent bases;
- $\gamma(k)$ = destabilization free-energy of an interior loop of k bases;
- $\beta(k)$ = destabilization free-energy of a bulge of k bases.

Zuker, like Waterman, assumed that Biophysics r ather than counting and maximizing the number of base pairs dictate RNA folding. He developed the energy minimization algorithm (dynamic programming algorithm), which assigns free energy scores to all possible structures. The structure with the lowest equilibrium free energy ΔG is the correct one. The following rules are used when calculating ΔG:

1. Stacking contributions are added from stems, not individual base pair contributions.
2. Energies are assigned to loops, stems, and all the structural elements, and are added up to give the overall ΔG value of the structure.
3. Pseudoknots are not taken into account.

The minimum free-energy $h(i,j)$ is the minimum of:

(*a*) Loop: $\alpha(a_i, a_j) + \xi(j - i - 1)$,

(*b*) Helix extension: $\alpha(a_i, a_j) + \eta + h(i+1, j-1)$,

(*c*) Bulge: $\min_{k \geq 1}\{\alpha(a_i, a_j) + \beta(k) + h(i+k+1, j-1)\}$,

(*d*) Bulge: $\min_{k \geq 1}\{\alpha(a_i, a_j) + \beta(k) + h(i+1, j-k-1)\}$,

(*e*) Interior loop: $\min_{k_1, k_2 \geq 1}\{\alpha(\alpha_i, \alpha_j) + \gamma(k_1 + k_2) + h(i + k_1 + 1, j - 1 - k_2)\}$.

Figure 2.5 illustrates an example ΔG calculation for an RNA stem loop (the overall ΔG = -4.6 kcal/mol).

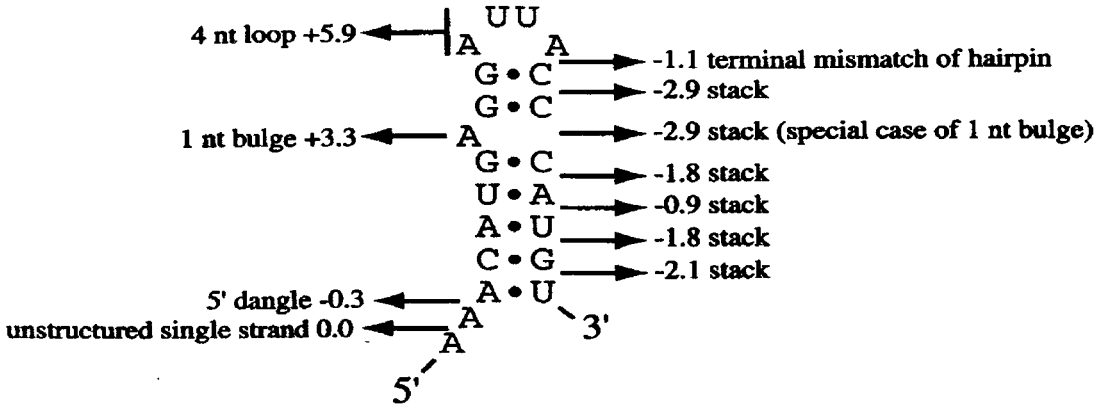

Figure 2.5: An example ΔG calculation for an RNA stem loop (Adopted from Durbin1998)

The minimum energy structure can be calculated recursively by a dynamic programming algorithm, similar to how the maximum base-paired structure was calculated in the Nussinov algorithm except the scores in the scoring matrix representing the minimum energy ΔG.

The algorithm starts a systematic search in all sub-fragments for the lowest free energy structure containing at least one base pair. The first pass will calculate the minimum free energy structures of all possible subsequences of length 5; the second pass will use these previously calculated values to find the minimum free energy of all subsequences of length 6. The algorithm is iterated until the minimum free energy structure of the sequence has been calculated. The lowest free energy structures are calculated, for each fragment, with and without the constraint that the terminal nucleotides are paired, and stored in the matrices.

The algorithm is summarized as follow:

Initialization:

$h(i, i\text{-}1) = 0$ for $i = 2$ to L;

$h(i, i) = 0$ for $i = 1$ to L;

Recursion:

$$h(i,j) = \min \begin{cases} \alpha(\alpha_i, \alpha_j) + \eta + h(i+1, j-1), \\ \min_{k \geq 1} \{\alpha(\alpha_i, \alpha_j) + \beta(k) + h(i+k+1, j-1)\}, \\ \min_{k \geq 1} \{\alpha(\alpha_i, \alpha_j) + \beta(k) + h(i+1, j-k-1)\}, \\ \min_{k_1, k_2 \geq 1} \{\alpha(\alpha_i, \alpha_j) + \gamma(k_1 + k_2) + h(i+k_1+1, j-1-k_2)\}. \end{cases}$$

In order to find one of the minimum free energy structures, we recover a traceback in the matrix similar as what we do in the Nussinov traceback algorithm.

2.3 Stochastic context-free grammars

2.3.1 Context-free grammars for modeling RNA

A context-free grammar G consists of a set of nonterminal symbols N, a terminals alphabet Σ, a set of productions P, and the start symbol S_0.

Every context-free grammar (CFG) production has the form $S \rightarrow \alpha$ where $S \in N$ and $\alpha \in (N \cup \Sigma)^*$ (the symbol * means zero or one or more occurrences of the letters in the alphabet set), thus the left-hand side consists of one nonterminal and there is no restriction on the number or placement of nonterminals and terminals on the right-hand side.

The production $S \rightarrow \alpha$ means that the nonterminal S can be replaced by the string represented by α.

We say the string β can be derived from α if there exists a sequence of direct derivations $\alpha_0 \rightarrow \alpha_1$, $\alpha_1 \rightarrow \alpha_2$, ..., $\alpha_{n-1} \rightarrow \alpha_n$ such that $\alpha_0 = \alpha$ and $\alpha_n = \beta$. Thus a derivation is an order of productions to generate a string.

Figure2.6 depicts a simple context-free grammar to parse a RNA sequence and thus obtain its secondary structure. For the RNA sequence CAUCAGGGAAGAUCUCUUG, the grammar whose productions are given in Figure 2.6(a) yields the parse tree of Figure 2.6(c), which reflects the specific secondary structure in Figure 2.6(d). Figure 2.6(b) lists the derivation of the production rules.

Productions (in Figure2.6(a)), which have the form: $S \rightarrow \alpha S \alpha$, *where* α denotes one of the terminals in the terminal alphabet (in case of RNA modeling, the terminal alphabet Σ includes A, C, G, and U), represent the base-pairings.

Productions, which have either the form: $S \rightarrow \alpha\, S$ *or* $S \rightarrow S\, \alpha$, represent the loops. The productions ($S_6 \rightarrow$ A S_7, $S_7 \rightarrow$ G S_8, *and* $S_8 \rightarrow$ G) in Figure 2.6(a) generate the left loop in Figure 2.6(d).

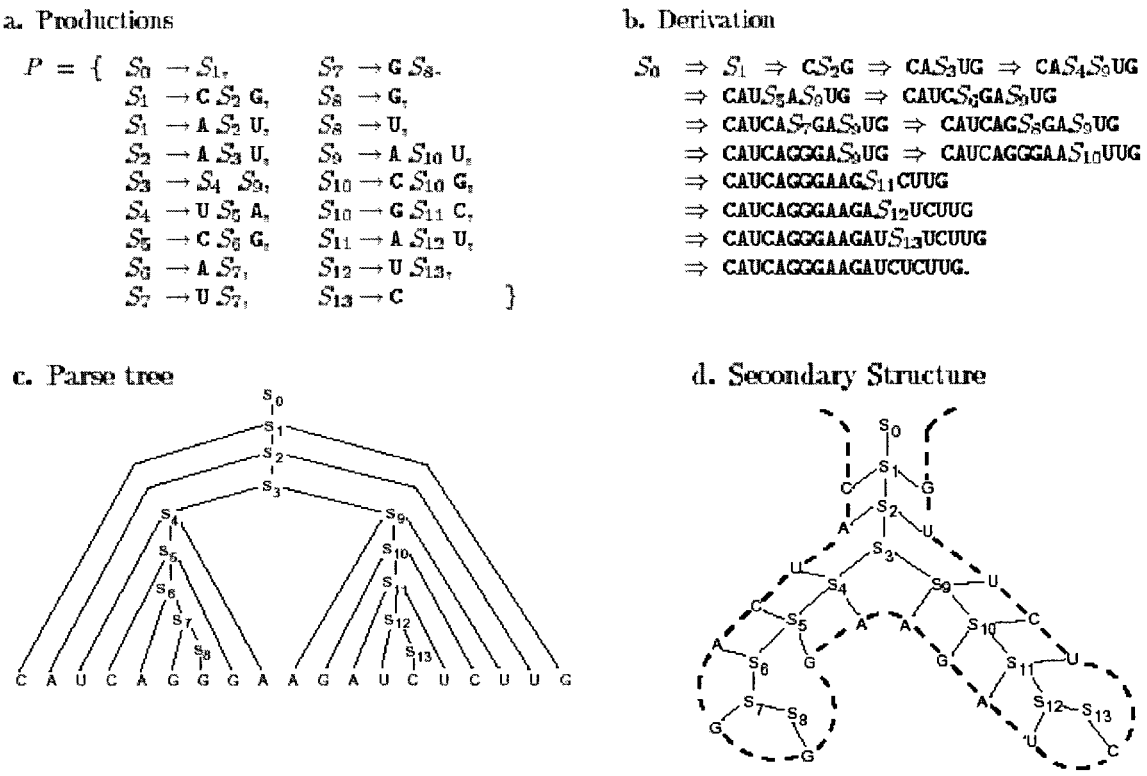

Figure 2.6: A simple context-free grammar for RNA structure prediction (Adopted from Sakakibara1994)

2.3.2 Stochastic context-free grammars

Stochastic context-free grammars (SCFGs) can be regarded as an extension of Hidden Markov models (HMMs, Krogh1994) that capture pairwise interactions in RNA secondary structures. SCFG can also be viewed as a probabilistic language, which can take in a string, a RNA sequence in case of RNA secondary structure prediction, and generate its probability of how likely the sequence belongs to the sequence family modeled by the SCFG.

We can obtain a SCFG by assigning probability values on the corresponding context-free grammar's production rules (if we assign probability to every production rules in Figure 2.6(a), we obtain a SCFG). So, in a SCFG, every production for a nonterminal *S* has an associated probability value such that a probability distribution exists over the set of productions for *S* (any production with the nonterminal *S* on the left side is called "a production for *S*").

SCFG generates sequences and assigns a probability to each generated sequence, and hence defines a probability distribution on the set of sequences that satisfies:

$\sum$ (Probabilities of all derivations/parse trees for all sequences) = 1.

The probability of a derivation *d* (parse tree) for a sequence *s* given a stochastic context-free grammar *G* is the product of the probabilities of the production instances applied to produce the derivation. Let $\Pr ob(S_0 \overset{d}{\Rightarrow} s|G)$ to denote the probability of a derivation *d,* it can be calculated using the following equation:

$$\Pr ob(S_0 \overset{d}{\Rightarrow} s|G) = \Pr ob(S_0 \Rightarrow \alpha_1|G).\Pr ob(\alpha_1 \Rightarrow \alpha_2|G).....\Pr ob(\alpha_n \Rightarrow s|G).$$

It satisfies: $0 \leq \Pr ob(S_0 \overset{d}{\Rightarrow} s|G) \leq 1.$

The probability of a sequence *s* is the sum of probabilities over all possible derivations that SCFG could use to generate *s*. Let Prob(*s* | *G*) denote the probability of a sequence *s,* it can be calculated by the following equation:

$$\text{Prob}(s \mid G) = \sum_{\substack{all\ \ derivations \\ (or\ \ parse\ \ trees)d}} \Pr ob(S_0 \overset{d}{\Rightarrow} s|G)$$

$$= \sum_{\alpha_1 \cdots \alpha_n} \Pr ob(S_0 \Rightarrow \alpha_1 | G).\Pr ob(\alpha_1 \Rightarrow \alpha_2 | G).....\Pr ob(\alpha_n \Rightarrow s | G).$$

There are three major tasks needed to be solved when using SCFGs to predict RNA secondary structure.

First one is to find an optimal alignment/parsing for a sequence, in which the sequence can be assigned a highest probability than others.

Given a sequence *s*, we want to find the most probably parse tree (structure), which is the parse tree with the highest probability. For a given sequence, there may be an exponential number of parse trees. To find the most probable parse tree, we need to generate all parse trees of *s* and then select the one with the highest probability. Such search method is very slow given the exponential number of parse trees for *s*.

The second one is to calculate the probability of a sequence given a SCFG.

The last one is to estimate the parameters of a SCFG given a set of training sequences.

The Cocke-Younger-Kasami (CYK) algorithm (Moll1988) was proposed to solve the first task. The second task can be solved using the Inside algorithm (Lari1990). The last task can be performed using the Inside-Outside algorithm (Lari1990), which is a variant of the EM (Expectation-Maximization) algorithm.

Once a SCFG model is constructed; it can be used to find the secondary structure of a new sequence; it can be used to discern general sequences from similar-length sequences of other kinds; it can also be used to produce multiple alignments of large sets of sequences.

Sakakibara (Sakakibara1994) used SCFGs to fold, model and align a family of tRNA sequences. The Expectation Maximization (EM) techniques were used to refine the parameters of a SCFG. The EM algorithm is summarized as follow:

1. Assigning initial values to the transition probability (the probability of transiting from one nonterminal to another nonterminal) and the symbols emission probabilities (the probability of emitting a terminal symbol in the terminal alphabet set Σ) of states to generate an initial grammar G_0.

2. Use G_0 and the CYK algorithm, to parse the raw sequences, producing a tree representation for each sequence indicated as T_{old}.

3. Use the grammar re-estimator algorithm to iteratively re-estimate the parameters until they stabilize and thus get a new grammar.

4. Use the new grammar to parse the sequences and thus create a new set of trees indicated as T_{new}.

5. Compare T_{old} and T_{new}, if they are significantly different then go to step 3, or stop.

Brown (Brown1995) developed a SCFG model, which is obtained through intersections of two stochastic context-free grammars, to model RNA pseudoknot structure. The model can discriminate RNA sequences, which contain a pseudoknot structure, from other RNA sequences not containing this structure.

Grate (Grate1994) applied SCFGs to predict long RNA sequences. The algorithm employed an adjustable minimum length encoding function to quickly identify potential base paired regions. It also utilized a divide-and-conquer approach to limit the amount of searching.

Knudsen (Knudsen1999) applied SCFGs incorporating evolutionary history to predict RNA secondary structure. The model uses maximum likelihood estimation to find the phylogenetic tree relating the sequences. The input of the SCFG is an alignment of RNA sequences and the output is a single common structure for the sequences.

Rivas (Rivas1999) presented the pseudoknot grammar, which encompasses the context-free grammar, to solve the problem of RNA secondary structure prediction with pseudoknot. The key feature of the grammar is the use of special non-terminal symbols, which dictate specific rearrangements of sub strings in a derivation. The algorithm has a worst-case complexity of $O(N^6)$ in term of time and $O(N^4)$ in term of space, for a string of length N.

The next three subsections summarize the above algorithms.

2.3.3 Optimal alignment determination

As mentioned above, we can use the CYK algorithm to parse a sequence and then to find an optimal alignment for the sequence given a SCFG.

Let i *and* j denote the indices for symbol x_i and x_j of a given sequence x; v is used to denote an index for nonterminals; x denotes the given sequence; θ denotes the given SCFG.

Let $\gamma(i,j,v)$ denote $\log P(x, \pi/\theta)$ where π is the most probable parse tree, which has the highest probability, and $\tau(i,j,v)$ denote the traceback, through which the optimal alignment can be recovered by tracing back the three dimensional dynamic programming matrix, which was used to store the alignment scores.

We can use the following algorithm (Durbin1998) to calculate $\log P(x, \pi/\theta)$.

Initialization: for i =1 to L, v = 1 to M:

$\gamma(i,i,v) = \log e_v(x_i)$;

$\tau(i,i,v) = (0,0,0)$.

Iteration: for $i = 1$ to L-1, $j = i + 1$ to L, $v = 1$ to M:

$\gamma(i,j,v) = \max_{y,z} \max_{k=i\ldots j-1} \{ \gamma(i,k,y) + \gamma(k+1,j,z) + \log t_v(x_i)\}$;

$\tau(i,j,v) = \max_{(y,z,k)}, k = i\ldots j\text{-}1 \{ \gamma(i,k,y) + \gamma(k+1,j,z) + \log t_v(x_i)\}$.

Termination: $\log P(x, \pi \mid \theta) = \gamma(1,L,1)$

The following algorithm (Durbin1998) can be used to recover the optimal alignment on optimal parse tree (structure).

Initialization:

Push $(1,L,1)$ on stack.

Iteration:

Pop(i,j,v);

$(y,z,k) = \tau(i,j,v)$.

If $\tau(i,j,v) = (0,0,0)$, attach x_i as the child of v;

Else: Attach y,z to parse tree as children of v.

Push$(k+1, j, z)$

Push(i, k, y).

2.3.4 Sequence probability computation

In order to use the Inside algorithm or the Outside algorithm to calculate the probability of a sequence, one should be aware that the Inside or Outside algorithm are only suitable for those SCFGs in which the production rules are restricted to 'Chomsky normal form'.

In Chomsky normal form, the production rules have the form of either $Wv \rightarrow WyWz$ or $Wv \rightarrow a$, where Wv, Wy, and Wz are nonterminals; a represents a terminal in the terminal alphabet (in case of RNA, the terminal alphabet consists of A, C, G, and U).

2.3.4.1 Inside Algorithm

Let $\alpha(i, j, v)$, called inside probability, stand for the probability of a parse subtree rooted at nonterminal Wv for subsequence $x_i,\ldots,x_j$ for all i, j, and v. Figure 2.7 shows an example of such subtree.

The $\alpha(i, j, v)$ is calculated recursively by summing parse subtrees for states y and z and smaller subsequences i to k and $k + 1$ to j, for all y, z, and k, weighted by the transition probability $W_v \rightarrow W_y W_z$.

So, the probability of a sequence can be calculated through a recursive style from the subsequence of length 1 to length L, where L is the length of the sequence.

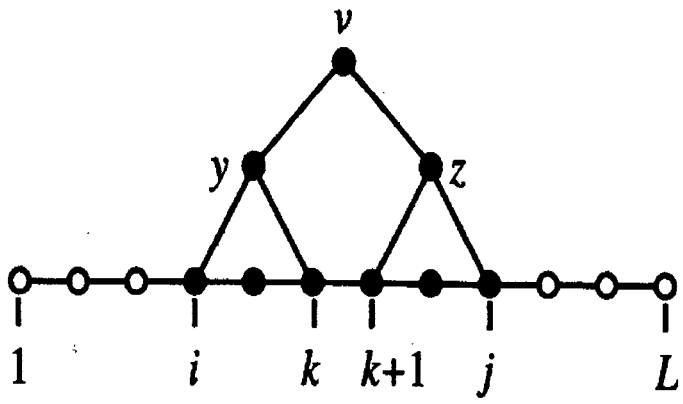

Figure 2.7: An example of a parse subtree rooted at nonterminal υ for the Inside algorithm (Adopted from Durbin1998)

The following summarizes the Inside algorithm (Durbin1998):

Initialization: for $i = 0$ to L, $v = 1$ to M:

$$\alpha(i, i, v) = e_v(x_i).$$

Iteration: for $i = 1$ to $L - 1, j = i + 1$ to L, $v = 1$ to M:

$$\alpha(i, j, \upsilon) = \sum_{y=1}^{M} \sum_{z=1}^{M} \sum_{k=i}^{j-1} \alpha(i,k,y)\alpha(k+1,j,z)t_{\upsilon}(y,z).$$

Termination:

$$P(x \mid \theta) = \alpha(1,L,1).$$

2.3.4.2 Outside Algorithm

Let $\beta(i, j, v)$, called outside probability, stand for the probability of a parse tree rooted at the beginning of the start nonterminal S for a sequence x, excluding all parse subtrees for subsequence $x_i,\ldots,x_j$ rooted at nonterminal Wv for all i, j, and v.

Figure 2.8 shows the recursive calculation of $\beta(i, j, v)$, the summed probabilities of all parse trees excluding subtrees rooted at nonterminal v, which generate the subsequence i, j(Durbin1998). Diagram (a) corresponds to the first part of the outside iteration equation for the contributions to $\beta(i, j, v)$ of combining the outside value for nonterminal y and subsequence $1, \ldots, k-1, j+1, \ldots, L$, the inside value for nonterminal z filling in the subsequence $k, \ldots, i-1$, and the transition probability for $y \rightarrow vz$. Diagram *(b)* corresponds to the second part of the iteration equation, which combines the outside probability for nonterminal y on the excluded subsequence $i, \ldots, k$, and the transition probability for $y \rightarrow zv$.

Similarly, the probability of a sequence x can be calculated through a recursive style from the largest excluded subsequence $x_1,\ldots,x_L$ to length 1, where L is the length of the sequence. The following summarizes the Outside algorithm (Durbin1998):

Initialization:

$$\beta(1,L,1) = 1;$$

$$\beta(1,L,v) = 0 \quad \text{for } v = 2 \text{ to } M.$$

Iteration: for $i = 1$ to L, $j = L$ to i, $v = 1$ to M:

$$\beta(i,j,v) = \sum_{y,z}\sum_{k=1}^{i-1} \alpha(k,i-1,z)\,\beta(k,j\,y)t_y(z,v)$$

$$+ \sum_{y,z}\sum_{k=j+1}^{L} \alpha(j+1,k,z)\,\beta(i,k,y)t_y(v,z)$$

Termination:

$$P(x/\theta) = \sum_{v=1}^{M} \beta(i,i,v)\, e_v(x_i) \text{ for any } i.$$

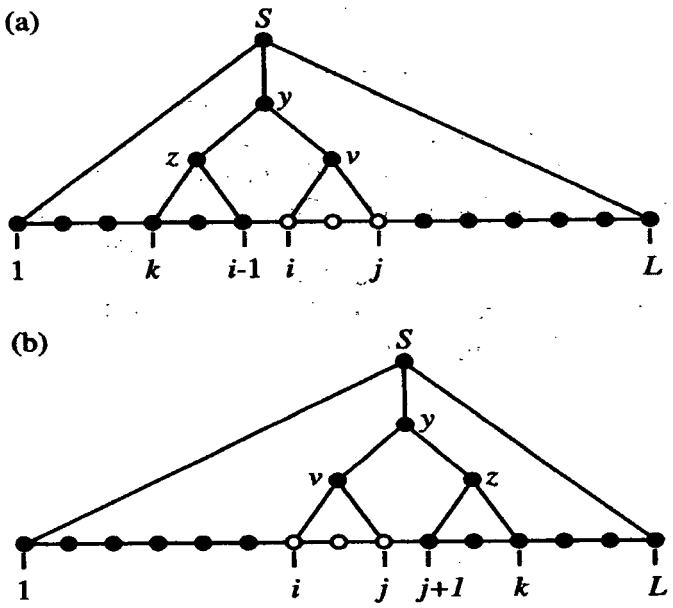

Figure 2.8: An example of a parse subtree rooted at nonterminal υ for the Outside algorithm (Adopted from Durbin1998)

2.3.5 Parameters estimation

Similarly to the Forward-Backward algorithm (Durbin1998) used in HMMs (Krogh1994), a variation of the EM techniques can be used in the Inside-Outside algorithm (Lari1990) to refine the parameters of a SCFG.

We use the term $c(v)$ to denote the number of times that state v is used in a derivation, and

$$c(v) = \frac{1}{P(x|\theta)} \sum_{i=1}^{L} \sum_{j=i}^{L} \alpha(i,j,v)\beta(i,j,v) . \quad 2.1.$$

The term $c(v \rightarrow yz)$ is used to denote the number of times that production rule $Wv \rightarrow WyWz$ is used in a derivation, and

$$c(\upsilon \rightarrow yz) = \frac{1}{P(x|\theta)} \sum_{i=1}^{L-1} \sum_{j=i+1}^{L} \sum_{k=i}^{j-1} \beta(i,j,\upsilon)\alpha(i,k,y)\alpha(k+1,j,z)t_{\upsilon}(y,z) \quad 2.2.$$

The term $t'_{\upsilon}(y,z)$ is used to denote the probability of production rule $W\upsilon \rightarrow WyWz$, and

$$t'_{\upsilon}(y,z) = \frac{c(\upsilon \rightarrow yz)}{c(\upsilon)} = \frac{\sum_{i=1}^{L-1}\sum_{j=i+1}^{L}\sum_{k=i}^{j-1} \beta(i,j,\upsilon)\alpha(i,k,y)\alpha(k+1,j,z)t_{\upsilon}(y,z)}{\sum_{i=1}^{L}\sum_{j=i}^{L}\alpha(i,j,\upsilon)\beta(i,j,\upsilon)} \quad 2.3.$$

The term $\hat{e}_{\upsilon}(a)$ is used to denote the emission probability of a symbol generated by the production rule $W\upsilon \rightarrow a$, and

$$\hat{e}_{\upsilon}(a) = \frac{c(\upsilon \rightarrow a)}{c(\upsilon)} = \frac{\sum_{i|x_i=a} \beta(i,j,\upsilon)e_{\upsilon}(a)}{\sum_{i=1}^{L}\sum_{j=i}^{L}\alpha(i,j,\upsilon)\beta(i,j,\upsilon)} \quad 2.4.$$

The EM (Inside-Outside) algorithm is summarized as follow (Durbin1998):

Initialization: select arbitrary SCFG parameters.

Repetition

For each sequence $j = 1 \ldots n$:

Use Inside algorithm to calculate $\alpha(i,j,\upsilon)$.

Use Outside algorithm to calculate $\beta(i,j,\upsilon)$.

Add the contribution of sequence j to equation 2.1 and equation 2.2.

Calculate the new SCFG parameters using equation 2.3 and equation 2.4.

Calculate the probability of the training sequences of the SCFG.

Termination: if the probability of the sequences is stable according to some criteria (example: the summed probability of all sequences does not change significantly).

However, we should indicate that a SCFG with too many free parameters cannot be estimated well from a relatively small set of training sequences. Sometimes it will run into a situation where the SCFG fits the training sequences well but gives poor results on a related sequence that is not included in the training sequence set. This phenomenon is called over-fitting (Eddy1994).

One of the effective methods to deal with such problem is to control the number of free parameters using prior information. This can be overcome through maximum a posterior (MAP) estimation (Eddy1994). Another way is to use as many sequences as we can obtain to train the model.

2.3.6 Advantages and disadvantages of SCFGs

SCFGs can capture the secondary structure of RNA, and they have been successfully used on the secondary structure prediction of the tRNA. SCFG is one of the best methods known to predict the secondary structure of RNA molecules.

However, all of the memory complexities of the CYK algorithm (for the optimal alignment), the Inside algorithm (for calculating the probability), and the Inside-Outside algorithm (for the parameter estimation) are $O(ML^2)$, and the time complexities of the CYK algorithm, the Inside algorithm, and the Inside-Outside algorithm are $O(L^3M^3)$, where L is the average length of the sequences and M is the number of terminals in the SCFG.

So, it is impractical to use SCFG to predict the secondary structure or other higher order structure of large length sequences such as rRNA. SCFG also cannot successfully resolve the problem of RNA secondary structure prediction with pseudoknot, which can be thought of one type of the tertiary structure elements.

2.4 Covariance Models

Since our project is based on covariance models (CMs), which are equivalent to SCFGs, we take them out from the previous section and summarize their distinctive characteristic in the following.

A CM is based on an ordered tree that captures all pairwise interactions of an RNA secondary structure. Pairwise nodes of the tree are assigned to base pairs in RNA structure and singlet nodes are assigned to single-stranded bases. Figure 2.9 depicts a simple example of a RNA secondary structure (A) and its related ordered tree (B).

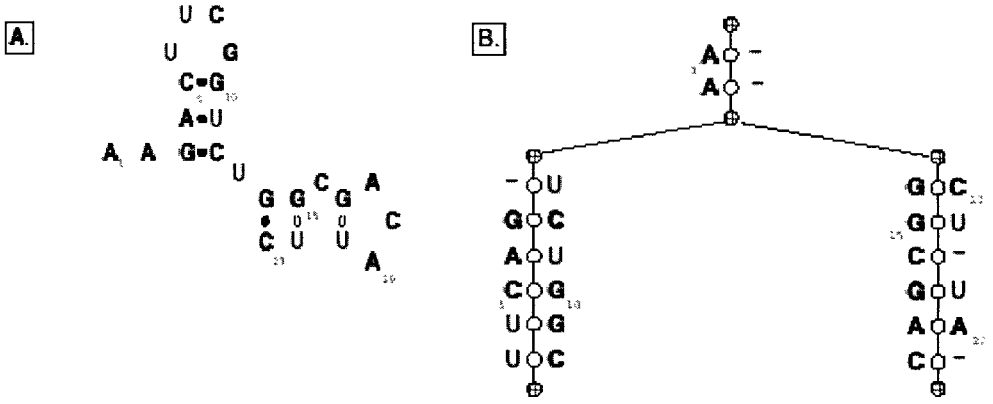

Figure 2.9: A simple example of a RNA secondary structure(A)
and its related ordered tree (B) (Adopted from Eddy1994).

A CM represents the consensus secondary structure of a multiple alignment, which belongs to the same sequence family modeled by the CM.

Figure 2.10 depicts the relationships among the RNA secondary structure (similar to the RNA secondary structure given in Figure 2.9), a multiple sequence alignment, and a simple way to represent the consensus secondary structure.

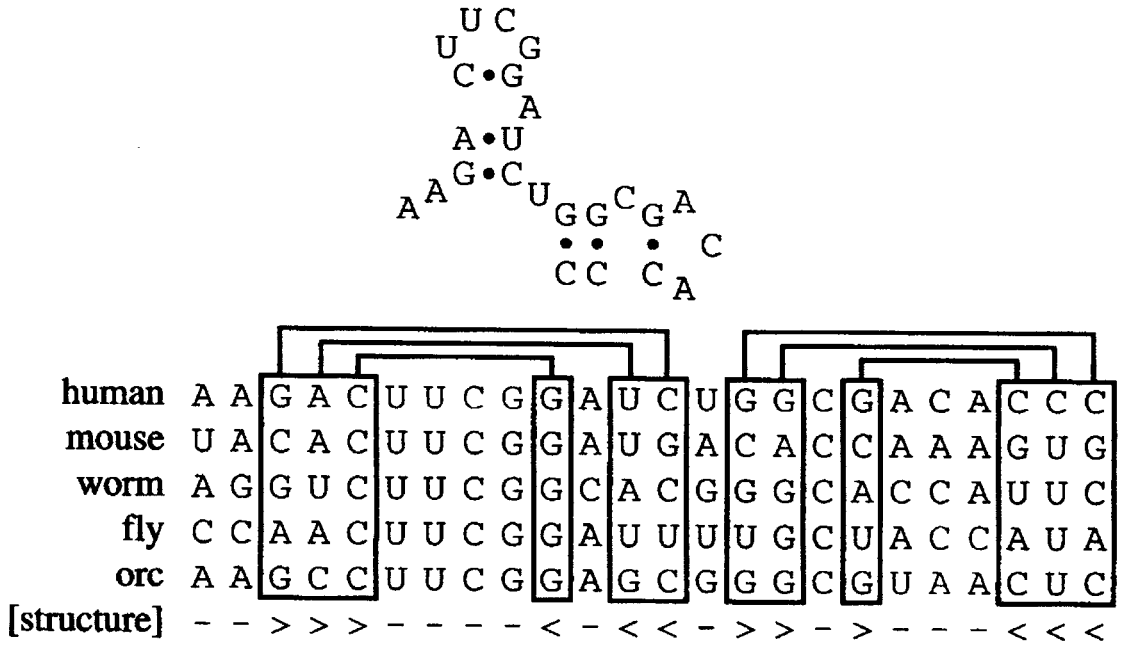

Figure 2.10: The relationships among the RNA secondary structure, a multiple sequence alignment, and a simple way to represent the consensus secondary structure (Adopted from Durbin1998).
Upper: RNA secondary structure for human;
Middle:a multiple sequence alignment;
Lower: a simple way to represent the consensus secondary structure

In a CM, each node describes columns in a multiple alignment instead of bases in an individual sequence. Specific base assignments are replaced with symbol emission probabilities assigned to 16 possible pairwise or 4 singlet nucleotides.

Figure 2.11 illustrates a simple CM and its related parsed tree for a secondary structure of a RNA segment.

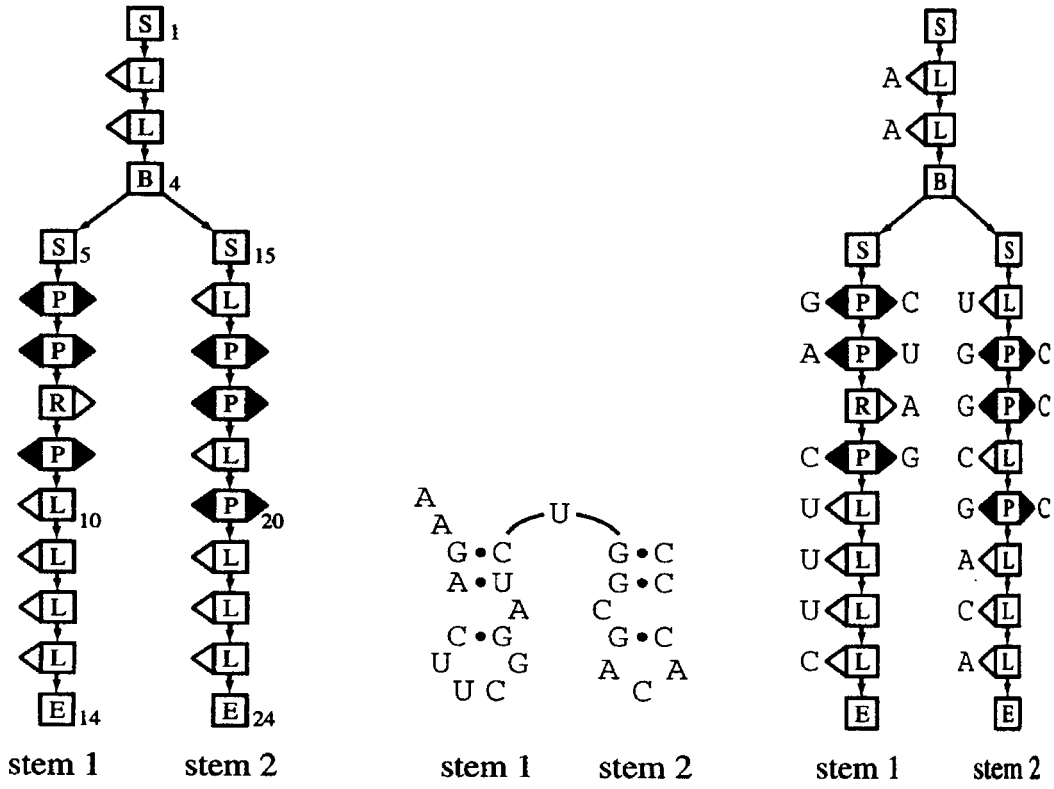

Figure 2.11: A simple CM and its related parsed tree for a secondary structure of a RNA segment (Adopted from Durbin1998).

Left: a CM representing the consensus secondary structure of the multiple alignment given in Figure 2.10; Middle: secondary structure of the first RNA sequence from given Figure 2.10;

Right: a shrinking-parsing tree for the sequence given in Middle (The folding corresponding to the secondary structure).

This CM represents the consensus secondary structure (left part of Figure 2.11) of a set of unfolded sequences (in Figure 2.10).

Figure 2.12 depicts the production rules used to produce the secondary structure (in the middle of Figure 2.11). It also demonstrates that CMs are fundamentally equivalent to SCFGs.

$$
\begin{array}{lcl}
 & & \textbf{Stem 1} \\
S_1 \rightarrow L_2 \ldots & S_5 \rightarrow P_6 & S_{15} \rightarrow L_{16} \\
L_2 \rightarrow aL_3 \ldots & P_6 \rightarrow gP_7c \ldots & L_{16} \rightarrow uP_{17} \ldots \\
L_3 \rightarrow aB_4 \ldots & P_7 \rightarrow aR_8u \ldots & P_{17} \rightarrow gP_{18}c \ldots \\
B_4 \rightarrow S_5S_{15} & R_8 \rightarrow P_9a \ldots & P_{18} \rightarrow gL_{19}c \ldots \\
 & P_9 \rightarrow cL_{10}g \ldots & L_{19} \rightarrow cP_{20} \ldots \\
 & L_{10} \rightarrow uL_{11} \ldots & P_{20} \rightarrow gL_{21}c \ldots \\
 & L_{11} \rightarrow uL_{12} \ldots & L_{21} \rightarrow aL_{22} \ldots \\
 & L_{12} \rightarrow cL_{13} \ldots & L_{22} \rightarrow cL_{23} \ldots \\
 & L_{13} \rightarrow gE_{14} \ldots & L_{23} \rightarrow aE_{24} \ldots \\
 & E_{14} \rightarrow \epsilon & E_{24} \rightarrow \epsilon
\end{array}
$$

Figure 2.12: Production rules used to produce the stem 1 and stem 2 given in Figure 2.11

(Adopted from Durbin1998)

Eddy used CM to model the consensus or common secondary structure of tRNA families. Eddy used a 3D dynamic programming algorithm, which is similar with the algorithm proposed by Nussinov and Zuker, to decide a structure for a model. That means to decide how many nodes and states and how they match a model.

Eddy also used expectation maximization (EM) algorithm to find optimal values for the state transition probabilities and the symbol emission probabilities of a model.

Figure 2.13 depicts a common structure of a CM proposed by Eddy (Eddy1994).

In this CM, each node is replaced with a number of different states to accommodate insertions and deletions.

Leftwise nodes for single-stranded consensus positions expand into match-left, insert-left and delete states represented by *MATL*, *INSL* and *DEL* respectively. Rightwise singlet nodes expand into match-right, insert-right and delete states represented by *MATR*, *INSR* and *DEL* respectively. Thus, leftwise nodes become a triplet of states *MATL*, *INSL*, and *DEL*. Rightwise nodes become a triplet of states *MATR*, *INSR* and *DEL*.

The pairwise nodes expand into six states: an *MATP* state (for a base pairing), a *DEL* state (for complete deletion of the base pair), *MATL* and *MATR* states (for a single-base deletion that removes the 3' or 5' base, respectively), and *INSL* and *INSR* states that allow insertions on the 5' or 3' side of the pair, respectively.

The root start node is expanded to a begin state *BEG* and insert states for either the 5' or 3' side, represented by *INSL* and *INSR*. Bifurcation nodes and end nodes are represented by *BIF* and *END* respectively.

States are connected by arrows, which are assigned corresponding state transition probabilities.

For each pairwise state, there are 16 possible production rules $S \rightarrow xSx$. For each leftwise state, there are 4 possible production rules $S \rightarrow xS$. For each rightwise state, there are 4 possible production rules $S \rightarrow Sx$, where x denotes one of the symbols in the alphabet set {A, C, G, U}.

Because CMs are equivalent to SCFGs, a CM has the same advantages and disadvantages as SCFG.

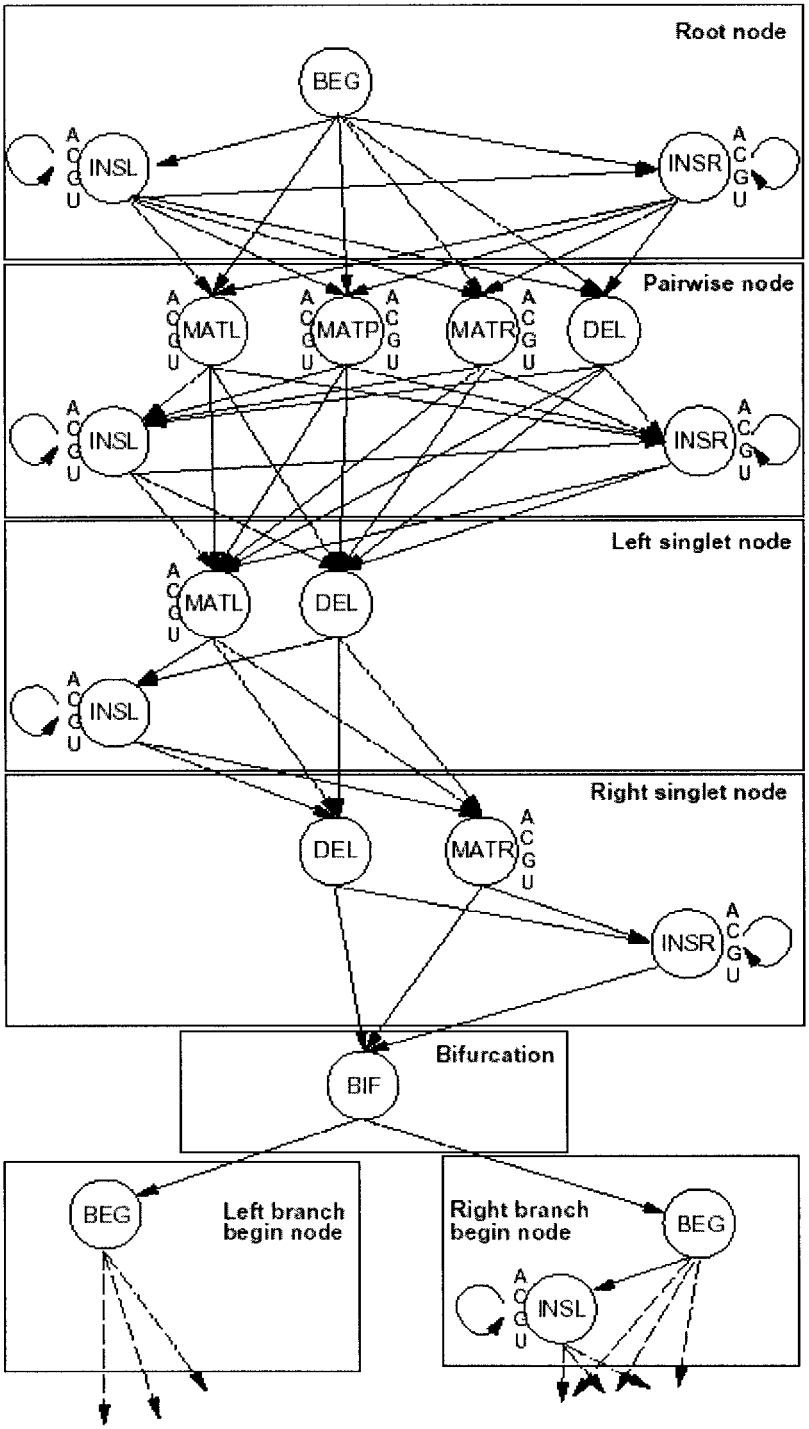

Figure 2.13: A typical structure of a CM (Adopted from Eddy1994)

2.5 Other methods

HMMs (Hidden Markov models) were also proposed to predict the secondary structure of RNA. HMMs correspond to stochastic regular grammar. They can capture the primary structure very well, but they cannot capture the secondary structure of RNA well due to the interactions between remote base pairs.

Genetic algorithm was proposed by Gultyaev (Gultyaev1995) to predict the secondary structure of RNA.

2.6 Limitation of the existing methods

For the comparative sequence analysis, there remains no reliable or automatic way of inferring an optimal consensus secondary structure even if the related sequences are already aligned. Because considerable manual intervention is still required to identify potential helices that maintain base complementarities.

For the minimization of free energy method, there are one or two significant problems associated with it. First, the energy parameters are inevitably imprecise. Hence, the true minimum free energy structure might be one that is suboptimal with respect to the parameters used. The same might hold because of unknown biological constraints that may change relative energies, turning an otherwise suboptimal structure into the most favorable one. Also under physiological conditions RNA sequences may exist in alternative states whose energy differences is small.

SCFG cannot be used to predict the secondary structure of large length sequences such as mRNA and rRNA. They also cannot successfully resolve the problem of RNA secondary structure prediction with pseudoknot, which can be thought of one type of the tertiary structures.

3 TRAINING SCFG IN PARALLEL

Parallel processing has made tremendous impact on many areas of computer application. With the raw computing power of parallel computers, it is possible to address many applications that were beyond the capability of conventional sequential computing techniques (Kumar1994).

Yap applied parallel algorithms on biological sequence analysis especially the multiple sequence alignment problem using speculative computation (Yap1995, Yap1998). Schmidt used parallel algorithms on protein database searching (Schmidt2002). Bokhari applied parallel techniques to implement the dynamic programming algorithm for DNA sequence alignment (Bokhari2003). Xiao applied parallel algorithm on a cluster of workstation for gene clustering (Xiao2003).

Message passing is a paradigm used widely on certain classes of parallel machines especially those with distributed memory. Over the last ten years substantial progress has been made in casting significant applications in this paradigm.

The goal of the MPI (Message Passing Interface) is to develop a widely used standard for writing message-passing programs (Snir1998, Gropp1998). As such the interface establishes a practical portable, efficient, and flexible standard for message passing.

MPI offers an application-programming interface not necessarily for compilers or a system implementation library.

MPI allows efficient communication and at the same time allows for implementations that can be used in a heterogeneous environment.

So, MPI is an ideal mechanics to reduce the execution time of the sequential SCFG training for RNA secondary structure prediction.

In this thesis, we apply MPI to the problem of tRNA secondary structure prediction. We implement two different methods to train SCFGs, one is the Parallel Single-SCFG Training model and another is the Parallel Multi-SCFG Training model.

The following sections describe these two methods.

3.1 Parallel Single-SCFG Training

In this method, we consider training a single SCFG in parallel.

As indicated earlier, in order to use SCFG to predict the tRNA secondary structure, we use the EM (Expectation Maximization) technique to refine the parameters of a SCFG model. In the EM method, we use the training sequences to refine the transition probabilities and emission probabilities in order to obtain accurate classification of unknown sequences.

In the previous methods, Eddy1994, the training sequences are parsed by the SCFG one by one in the EM iterative procedure. Figure 3.1 depicts the process of constructing the CM model using the sequential EM algorithm.

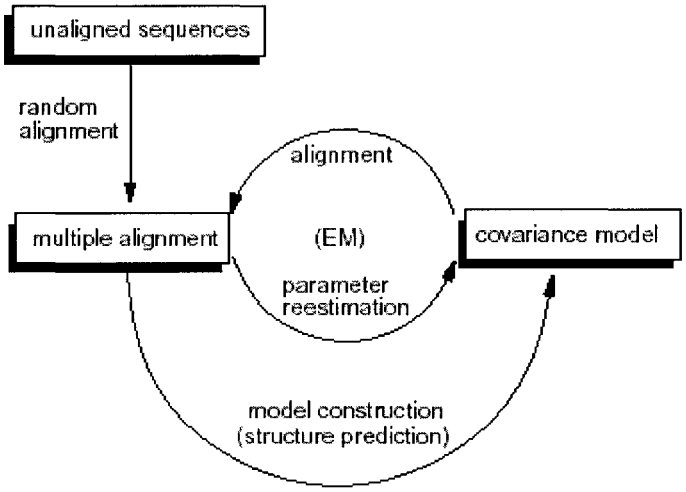

Figure 3.1: The process of constructing a CM model with sequential training in COVE

(Adopted from Eddy1994)

Obviously it is a time consuming process when the number of the training sequences is huge, such as 1000 training sequences, since the major time is consumed on the sequentially parsing/aligned process.

Figure 3.2 shows our Parallel Single-SCFG Training model. The detailed implementation and test results of this model will be given in chapter 4.

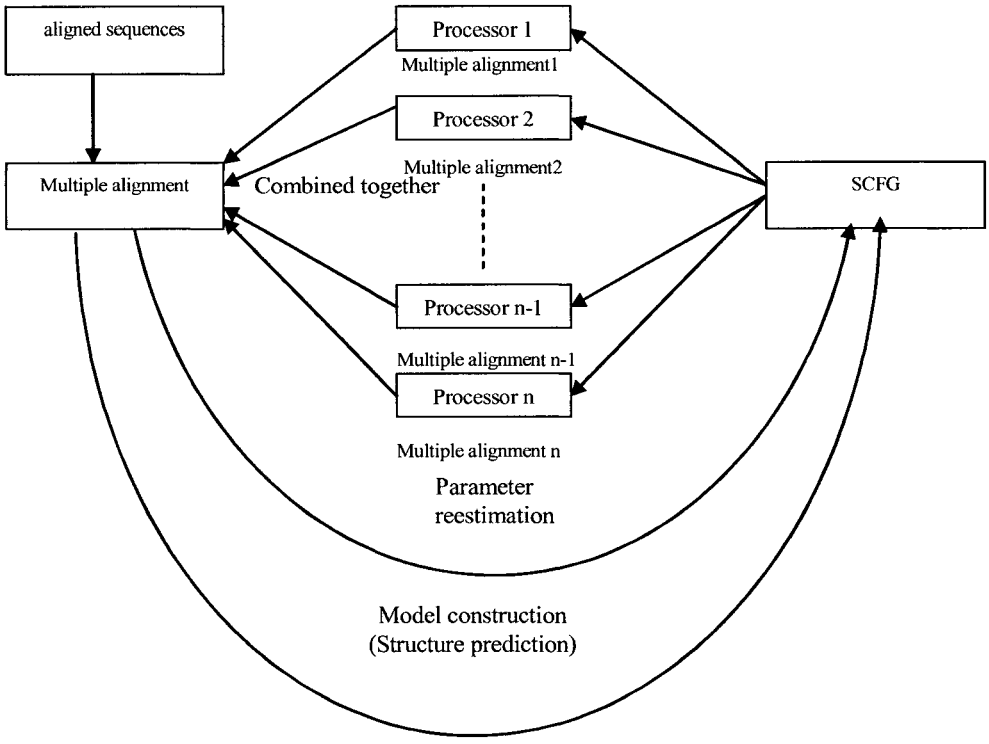

Figure 3.2: Parallel Single-SCFG Training model

In this model, we distribute the training data into different processors and obtain a multiple aligned/parsed sequences, then we gather all of these sequences together into a single multiple alignment and use them to construct a new SCFG.

The process will be iterated until the SCFG is stable according to some given convergence criteria.

The following summarizes our parallel single-SCFG training algorithm:

1. Constructing an initial SCFG for each processor;

We use an existing multiple alignment of tRNA sequences to construct an initial SCFG.

2. Parsing the training data in parallel;

We distribute the training data/sequences into different processors and parse/align them in parallel.

3. Combining aligned sequences into a single multiple alignment

We gather all of the parsed/aligned sequences from all processors together into a single new multiple alignment.

4. Constructing a new SCFG

We use the combined sequences to construct a new SCFG and go back to the second step if the new SCFG is unstable according to some convergence criteria (i.e. the summed parsing score for all of the training sequences changes little); otherwise we store the new SCFG and stop the training process.

3.2 Parallel Multi-SCFG Training model 1

A multiple alignment arranges a set of sequences in a scheme where positions that were believed to be homologous are written in a common column (Gribskov1989).

Profile analysis is a method for detecting distantly related sequences (such as proteins and RNA) by sequence comparison (Gribskov1989). The comparison information is expressed in a position-specific scoring table (profile), which is created from a group of sequences previously aligned by structural or sequence similarity. Comparing the target to the profile using dynamic programming algorithms can test the similarity of any other sequence to the group of aligned sequences.

In this model, each processor obtains a multiple alignment of sequences and uses it to generate an initial Sub-SCFG. Then each processor uses its own training data to train its Sub-SCFG (similar to the sequential training method in COVE) until it is stable according to given convergence criteria. After that, all of the Sub-SCFGs from all processors are combined into a single SCFG. Finally, we train the SCFG as in the parallel single-SCFG training model until it is stable according to the given convergence criteria.

The following summarizes the training process of this model.

1. We distribute the training data into different processors and use them to train initial Sub-SCFGs until they remain stable, same as the sequential training process in COVE (Eddy1994), using the EM technique.
2. We obtain a multiple alignment and its related consensus structure sequence from each processor after each Sub-SCFG is stable.

3. We combine all of the consensus structure sequences (which reflect a notion of shared or common perception of the secondary structure among a set of sequences) into a profile (which contains the information of the consensus structure alignment).
4. We combine the multiple alignments of all processors using the profile obtained in step 3 and then construct a new combined SCFG.
5. We use the whole training data to train the combined SCFG until it is stable according to the given convergence criteria.

Figure 3.3 illustrates the process of our Parallel Multi-SCFG Training model.

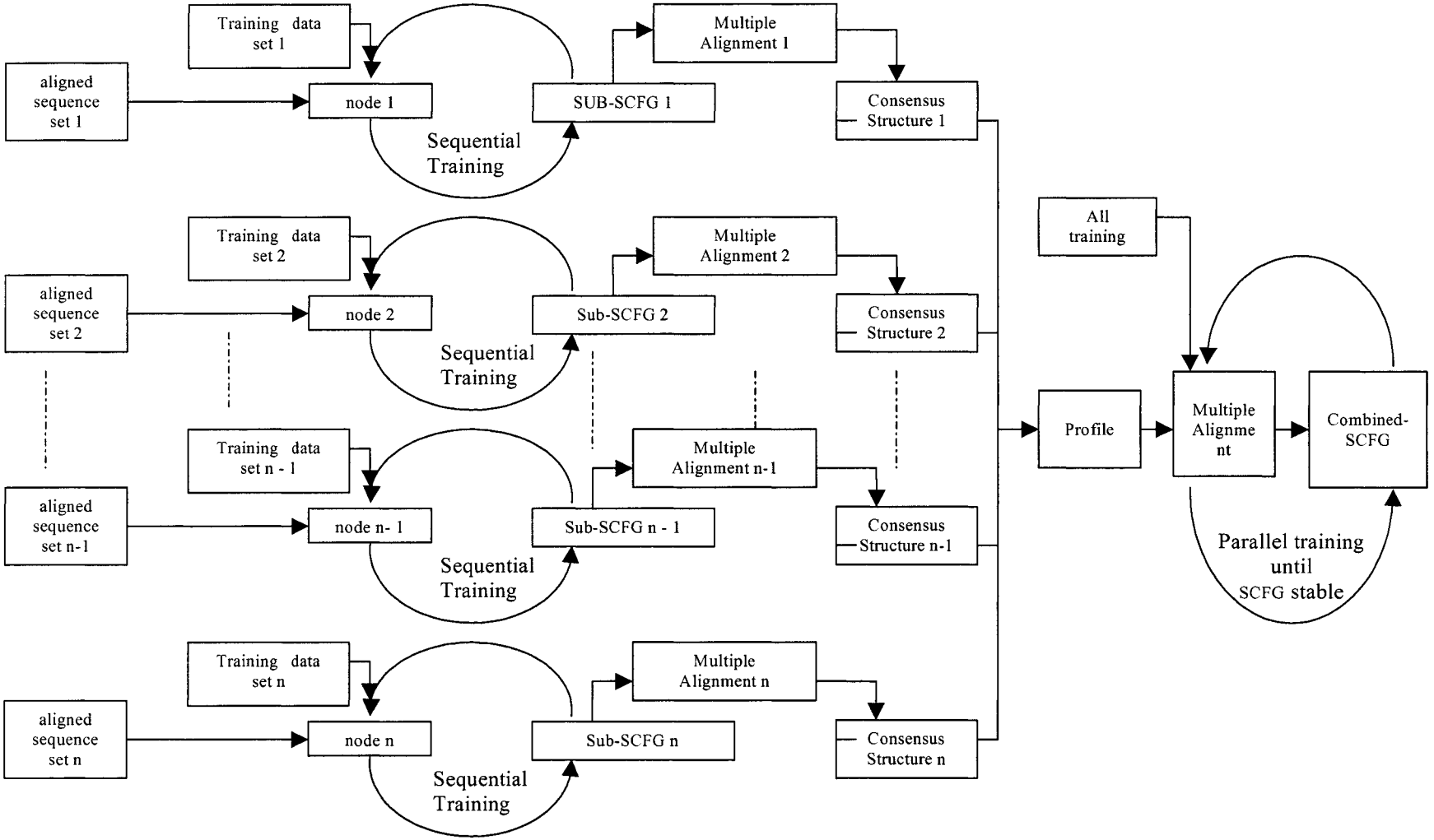

Figure 3.3: Parallel Multi-SCFG Training model 1

Since the lengths of the multiple-alignment and the consensus structure from each processor are different with each other, they cannot be straightforwardly combined into a big multiple alignment.

However, optimally aligning many multiple-alignment in term of time complexity is NP-complete, which means there is no known polynomial-time algorithm for finding a solution but a solution can be checked for correctness in polynomial time (Durbin1998).

Because the consensus structure of each Sub-SCFG conserves the motif information (such as AA-Stem, D-Arm, AC-Arm, V-Loop, and T-Arm for tRNA families) of the training data's secondary structure, the consensus structures from each Sub-SCFG can be aligned together to get a multiple alignment/profile of the consensus structure. This multiple alignment of consensus structures can be used to deduce the big multiple alignment of all the training data.

The novel method we use to obtain a multiple alignment for all of the training data is that the Center-Star like algorithm, embedded in which is the dynamic programming algorithm (Durbin1998), is used to heuristically combine the consensus structures into a profile. And then, using the profile to obtain a combined multiple alignment.

Section 3.3.1 and 3.3.2 explain these methods.

3.2.1 Pairwise dynamic programming algorithm for consensus structure alignment

An alignment of two sequences s and t is an arrangement of s and t by position, where s and t can be padded with gap symbols to achieve the same length. The following shows an example alignment of sequence s = AGCACACA and sequence t = ACACACTA:

$$\begin{array}{ll} s: & AGCACAC-A \\ t: & A-CACACTA \end{array} \quad \text{or} \quad \begin{array}{l} AG-CACACA \\ ACACACT-A \end{array}$$

Dynamic programming algorithms consist of solving an instance of a problem by taking advantage of already computed solutions for smaller instance of the same problem. We start with the shorter prefixes and use previously computed results to solve the problem for larger prefixes until the length of the prefixes are equal to the length of the entire sequences.

A prefix of a sequence S is any substring of S of the form $S_{1...j}$ for $0 \leq j \leq |S|$. We admit $j = 0$ and define $S_{1...0}$ as being the empty subsequence, which is a prefix of S as well.

Given two sequences S and T, with $|S| = n$ and $|T| = m$ (i.e. $S = S_{1...n}$ and $T = T_{1...m}$), there are $n + 1$ and $m + 1$ prefixes of S and T, respectively (including the empty string). The basis for $V(i, 0)$ says that if i characters of S are to be aligned with 0 characters of T, then they must all be matched with spaces. The basis for $V(0, j)$ is analogous.

We define $V(i, j)$ as the similarity/scoring value of an optimal alignment of $S_{1...i}$ and $T_{1...j}$. Let σ be the scoring function.

The optimal alignment of the prefixes $S_{1...i}$ and $T_{1...j}$, in particular observing the last aligned pair of characters in such an alignment, must be one of the following three cases:

1. $(S_i, -)$: The score in this case is the score $\sigma(S_i, -)$ of aligning S_i with a space plus the score $V(i\text{-}1, j)$ of aligning the prefixes $S_{1...i\text{-}1}$ and $T_{1...j}$;
2. (S_i, T_j): The score in this case is the score $\sigma(S_i, T_j)$ of aligning S_i with T_j plus the score $V(i\text{-}1, j\text{-}1)$ of aligning the prefixes $S_{1...i\text{-}1}$ and $T_{1...j\text{-}1}$;
3. $(-, T_j)$: The score in this case is the score $\sigma(-, T_j)$ of aligning a space with a T_j plus the score $V(i, j\text{-}1)$ of aligning the prefixes $S_{1...i}$ and $T_{1...j\text{-}1}$.

The optimal alignment of $S_{1...i}$ with $T_{1...j}$ chooses the one which among these three possibilities has the greatest similarity value.

We can arrange the calculations in an $(n + 1) \times (m + 1)$ array V where entry $V(i, j)$ contains the similarity/score between $S_{1...i}$ and $T_{1...j}$. We compute $V(i, j)$ for all possible values of i and j. We start from smaller i, j and increase them, filling in the table in a row-wise (or column-wise) manner. Finally, $V(n, m)$ is the required maximum similarity/score between S and T.

The pairwise dynamic programming algorithm is summarized as follows:

Input: σ, S and T
Output: similarity between S and T
For $i = 0$ to n do
  For $j = 0$ to m do
  begin

$$V[i,j] \leftarrow \max \begin{cases} V[i-1,j] + \sigma[S_i,-] \\ V[i-1,j-1] + \sigma[S_i,T_j] \\ V[i,j-1] + \sigma[-,T_j] \end{cases}$$

  end
Return $V[n, m]$.

In RNA secondary structure prediction problem, the symbols '>', '<' and '•' are used to annotate RNA structural alignment (Eddy1994 and Konings1989). Symbol '>' means the majority letters (A, C, G and U) of this column is complementary with another column represented by the symbol '<'. The symbol '•' is used to denote all other columns.

Figure 3.4 illustrates an example of a multiple alignment and its related consensus structure. In this figure, five example sequences from different organisms that adopt the

same structure are shown in the top of the picture. Base-paired positions in the alignment are boxed and base-paired partners are connected by lines. The last line is a consensus structure representation (in this Figure, the symbol '•' is replaced by the symbol "-").

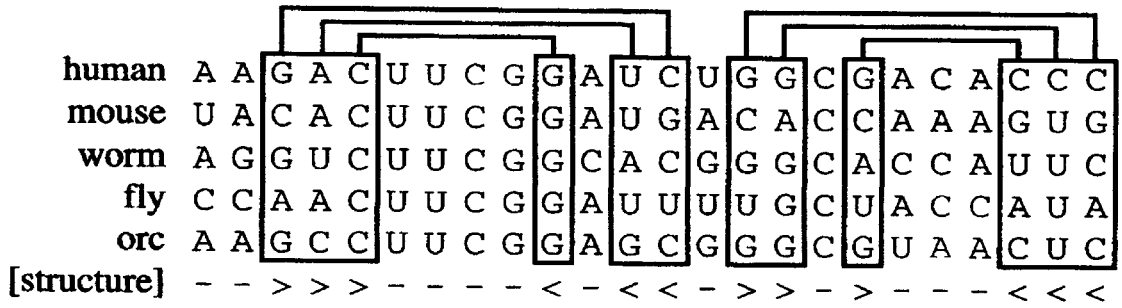

Figure 3.4: An example of a multiple alignment and its related consensus structure

(Modified from Durbin1998)

In order to use dynamic programming algorithm for the consensus structure alignment, we need to design a scoring function σ. Table 3.1 shows this scoring function represented by a matrix.

	>	<	•
>	4	-2	-1
<	-2	4	-1
•	-1	-1	1

Table 3.1: The scoring function (represented by the matrix) for the pairwise dynamic programming algorithm of the RNA consensus structure (alphabet set "> < •").

In order to give the priorities to the complementary consensus structures, we assign high scores to the matched symbols. Such as, in table 3.1, the score 4 is arbitrarily assigned to both the matched symbol pairs ('>' pair with '>' and '<' pair with '<').

We arbitrarily assign low scores to other base pairs, in the case of table 3.1, the score 1 is arbitrarily assigned to the pair ('•' pair with '•').

We also arbitrarily assign penalty scores to the mismatched pairs ('>' pair with '<', '<' pair with '>', '>' pair with '•', '•' pair with '>', '<' pair with '•', and '•' pair with '<'; see the table 3.1 for these scores).

The scoring function σ is summarized as follows:

σ (>, >) = 4, for the matched symbol pair: '>' pair with '>';
σ (>, <) = -2, the mismatched symbol pair: '>' pair with '<';
σ (>, •) = -1, for the mismatched symbol pair: '>' pair with '•';
σ (<, <) = 4, for the matched symbol pair: '<' pair with '<';
σ (<, >) = -2, for the mismatched symbol pair: '<' pair with '•';
σ (<, •) = -1, for the mismatched symbol pair: '<' pair with '•';
σ (•, •) = 1, for the matched symbol pair: '•' pair with '•';
σ (•, >) = -1, for the mismatched symbol pair: '•' pair with '>';
σ (•, <) = -1, for the mismatched symbol pair: '•' pair with '<'.

The driving forces that made us design such scoring function is that the consensus structure conserves the motif information for tRNA training data and the symbols '>' and '<' just represent such information about the motifs(see Figure 3.4).

Using this scoring function and the pairwise dynamic programming algorithm (Durbin1998), we can optimally align two consensus structure. For example: the sequences ">>>•<<<•" and "•>>>•••<<<•" can be aligned into aligned sequences:

Sequence 1 >>>•<<<• Aligned sequence 1: ->>>---•<<<•

Sequence 2 •>>>•••<<<• Aligned sequence 1: •>>>•••<<<•

The entry $V(M,N)$ of the score matrix stores the optimal alignment score, where M is the length of the second sequence and N is the length of the first sequence.

In the case of table 3.2, the optimal alignment score for the sequence ">>>•<<<•" and sequence "•>>>•••<<<•" is 38.

		>	>	>	•	<	<	<	•
	0	-4	-8	-12	-16	-20	-24	-28	-32
•	-4	-1	-5	-9	-11	-15	-19	-23	-27
>	-8	4	7	3	-1	-5	-9	-13	-17
>	-12	0	12	15	11	7	3	-1	-5
>	-16	-4	8	20	16	12	8	4	0
•	-20	-8	4	16	21	17	13	9	5
•	-24	-12	0	12	17	20	16	12	10
•	-28	-16	-4	8	13	16	19	15	13
<	-32	-20	-8	4	9	21	24	27	23
<	-36	-24	-12	0	5	17	29	32	28
<	-40	-28	-16	-4	1	13	25	37	33
•	-44	-32	-20	-8	-3	9	21	33	38

Table 3.2: A simple matrix for storing the scores of the pairwise alignment of the dynamic programming.

Sequence 1: ">>>•<<<•" is given in the first row;

Sequence 2: "•>>>•••<<<•" is given in the first column.

The optimal alignment can be recovered from the scoring matrix by trace-backing, starting on the entry $V(M,N)$ back to the entry $V(0, 0)$ determining which entries were responsible for the current one.

The trace-backing algorithm is summarized as follows:

Input: indices i, j, σ and matrix V of the alignment scores
Output: optimal alignment between S and T,
If $i = 0$ and $j = 0$ then
  $l \leftarrow 0$
Elseif $i > 0$ and $V[i, j] = V[i - 1, j] + \sigma(S_i, -)$ then
  Align($i - 1, j, l$)
  $l \leftarrow 1 + 1$
  align-$S[l] \leftarrow S[i]$
  align-$T[l] \leftarrow -$
Else
  if $i > 0$ and $j > 0$ and $V[i, j] = V[i - 1, j - 1] + \sigma(S_i, T_j)$ then
    Align($i - 1, j - 1, l$)
    $l \leftarrow l + 1$
    align-$S[l] \leftarrow S[i]$
    align-$T[l] \leftarrow - T[l]$
    Else
      If $j > 0$ and $V[i, j] = V[i, j - 1] + \sigma(-, T_j)$ then
        Align($i, j - 1, l$)
        $l \leftarrow l + 1$
        align-$S[l] \leftarrow -$
        align-$T[l] \leftarrow T[l]$

In the table 3.3, whose entries contain the traceback information which indicate where the score of the current entry come from, the optimal alignment of the sequence ">>>•<<<•" and sequence "•>>>•••<<<•" can be recovered and thus get the aligned sequences: ->>>---•<<<• and •>>>•••<<<•, where symbol '-' denotes a gap either in sequence 1 or in sequence2.

	0	1	2	3	4	5	6	7	8
0	0,0	0,0	0,1	0,2	0,3	0,4	0,5	0,6	0,7
1	0,0	0,0	0,1	0,2	0,3	1,4	1,5	1,6	0,7
2	1,0	1,0	1,1	1,2	2,3	2,4	2,5	2,6	2,7
3	2,0	2,0	2,1	2,2	3,3	3,4	3,5	3,6	3,7
4	3,0	3,0	3,1	3,2	4,3	4,4	4,5	4,6	4,7
5	4,0	4,1	4,2	4,3	4,3	5,4	5,5	5,6	4,7
6	5,0	5,1	5,2	5,3	5,3	5,4	5,5	5,6	5,7
7	6,0	6,1	6,2	6,3	6,3	6,4	6,5	6,6	6,7
8	7,0	7,1	7,2	7,3	7,4	7,4	7,5	7,6	8,7
9	8,0	8,1	8,2	8,3	8,4	8,4	8,5	8,6	9,7
10	9,0	9,1	9,2	9,3	9,4	9,4	9,5	9,6	10,7
11	10,0	10,1	10,2	10,3	10,3	10,5	10,6	10,7	10,7

Table 3.3: A matrix for storing the traceback information for the pairwise alignment of the sequences given in table 3.2.

Each entry contains the traceback information, which indicate where the score of the entry come from.

Since we are storing $(n + 1) \times (m + 1)$ numbers/scores, and each number costs us a constant number of calculations to compute(three sums and a max), the time complexity of the pairwise dynamic programming algorithm is *O(MN),* where *M* is the length of the second sequence and *N* is the length of the first sequence. The space complexity of the pairwise dynamic programming algorithm is also *O(MN).*

3.2.2 Multiple alignment of the consensus structures

Since optimally aligning multiple sequences is NP-complete[Durbin1998]. We cannot guarantee to obtain an optimal multiple alignment for the consensus sequences.

We use a Center-Star like heuristic algorithm to align the consensus structures. The basic idea of this algorithm is that we align the whole aligned consensus sequences to one of the unaligned consensus sequences each time until the entire unaligned consensus sequences have been aligned.

The algorithm consists of building a multiple sequence alignment *Mc* based upon the optimal global pairwise alignments between a fixed sequence *Sc* (the center) and all others. Each pairwise alignment is added to *Mc* using *Sc* as a guide.

One way to select the center sequence is to just try them all and then pick the best resulting score. Another way is to compute all optimal pairwise alignments and select as the center the string that maximizes

$$\sum_{i \neq c} \text{sim}(S_i, S_c).$$

Where $\text{sim}(S_i, S_c)$ stand for the similarity score between sequences S_i and S_c.

Let $\Sigma_{i \neq c} D(S_c, S_i)$ stand for consensus distance: the number of characters in all strings S_i that differ from the consensus string C: $\Sigma_i D(S_i, C)$. The term $D(S, T) = \sum_{i=1}^{l} \delta(S_i', T_i')$

denotes the score of an alignment of S and T, where S' and T' are variations of S and T with spaces inserted respectively.

The algorithm is summarized as follows:

Center Star algorithm:

Input: δ, $S = \{S_1, S_2, \ldots, S_k\}$

Output: optimal multiple alignment of $S_1, S_2, \ldots, S_k$

Find the center Sc such that $\Sigma_{i \neq c} D(S_c, S_i)$ is minimal

$S = \{S_1, S_3, \ldots, S_{k-1}\}$

For $i = 1$ to $k - 1$ do

    Find an optimal global alignment $[S'_c, S'_i]$ between S_c and S_i

Let $M_c = \{\text{best } S'_c,\}$

For $i = 1$ to $k - 1$ do

begin

    Add S'_i to M_c

    If needed, add spaces to all pre-aligned strings

end

Return M_c

Since we align the consensus structures one by one using the previous pairwise dynamic programming algorithm, the total number of times we apply the dynamic programming algorithm is:

$$1 + 2 + 3 +, \ldots, + N - 1, + N = N * (N + 1) / 2.$$

So the time complexity of the center star algorithm is $O(N^2L^2)$, where N is the number of the consensus sequences (or number of the processors, since every processor has one sub-

SCFG which can produce one consensus structure) and L is the average length of the consensus structures.

The time complexity can be decreased if we carefully select one sequence from the aligned consensus structures and align it with the new added unaligned sequence.

In this case, the total number of times we apply the dynamic programming algorithm is N - 1, where N is the number of the consensus sequences. So the time complexity will decrease to $O(NL^2)$.

3.3 Parallel Multi-SCFG Training model 2

Figure 3.5 depicts another Parallel Multi-SCFG Training model. The model is summarized as follows:

1. Each processor gets a multiple alignment of sequences and uses them to generate an initial Sub-SCFG.
2. We distribute the training data into different processors and use them to train the initial Sub-SCFGs until they are stable according to the given convergence criteria.
3. We obtain a multiple alignment and its related consensus structure sequence from each processor when each Sub-SCFG is stable.
4. We combine all of the consensus structure sequences into a profile.
5. We combine all of the multiple alignments obtained in all processors into a big multiple alignments and thus construct a combined SCFG.

3.4 Differences between our methods

The significant differences between the Parallel Single-SCFG Training (PSST) model and the Parallel Multi-SCFG Training (PMST) mode 1 are summarized as follow:

1. In the PSST model, each processor gets a copy of the SCFG and uses it to parse/align its own training data. After that, we combine the total parsed/aligned training data together and then construct a new SCFG.
2. In the PMST model 1, each processor gets a sub-SCFG and uses its own training data to train the sub-SCFG until it is stable. After that, we combine these sub-SCFGs together and then construct a new combined-SCFG. Finally, we use the whole training data to train the combined-SCFG in parallel until it is stable. This means we can get a better initial SCFG (may be a better grammatical structure).
3. The PMST model 1 can be used in a distributed system. That means we may obtain a combined SCFG from sub-SCFGs which may come from different computers in different countries.

The difference between the Parallel Multi-SCFG Training model 1 and the Parallel Multi-SCFG Training model 2 is that: if the combined SCFG is unstable (in step 5 of the PMST model 2), we use it to produce a multiple alignment for the entire training sequence and distribute the multiple alignment into different processors then iterate the training process from the first step.

Since both of the PMST model 1 and the PMST model 2 uses the same algorithms from step 1 to step 5, we implemented the PMST model 1 and the test results are given in chapter 4.

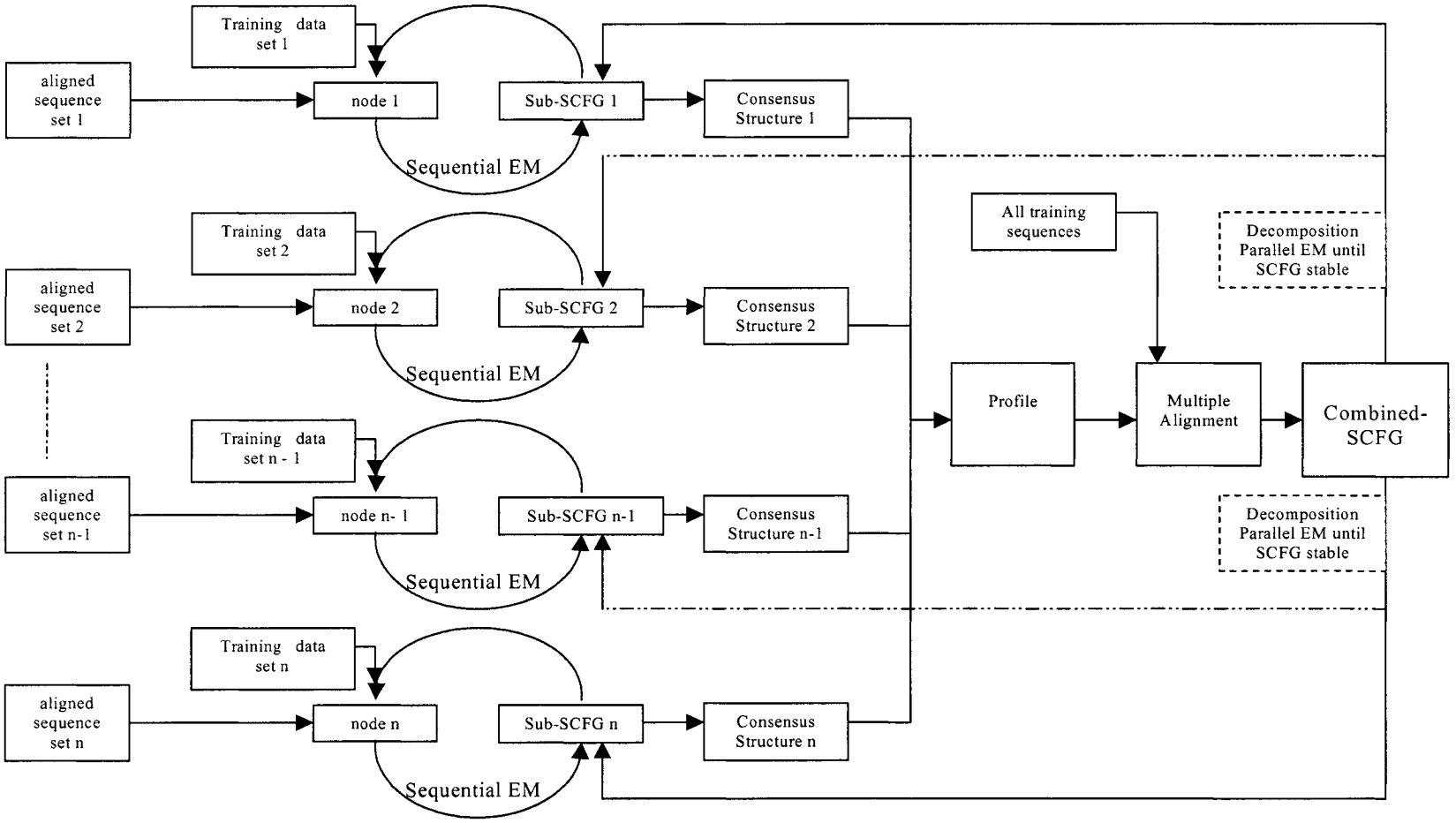

Figure 3.5: Parallel Multi-SCFG Training model 2

4 IMPLEMENTATION AND TEST

We have implemented the Parallel Single-SCFG Training model and the Parallel Multi-SCFG Training model using C language and MPI on Davinci, which is a parallel computer consisting of 12 processors at the Department of Computer Science, University of Windsor.

In order to check the scalability and the portability of our methods, the Parallel Single-SCFG Training model has been transferred and implemented on SHARC-NET.

These methods have been tested with tRNA family sequences with 1400 training data (these data are come from Sprinzl1996 which contains the known tRNA sequences of all organisms including organelles).

The following sections describe the detailed implementation and test results.

4.1 Parallel Single-SCFG Training model

In the SCFG construction process, we align/parse the training data in parallel until the SCFG is stable.

After the SCFG construction, we test the SCFG with different number of test data. The following sections summarize the SCFG construction process and test results.

4.1.1 SCFG construction

For the SCFG construction, we use an existing multiple alignment of tRNA sequences to construct an initial SCFG. Then we distribute the training sequences into each processor and gather all of the aligned sequences from them to construct a new SCFG. We iterate the above process until the SCFG is stable.

The following summarizes the process:

1. Constructing an initial SCFG in parallel;

We use a trust multiple alignment, which is obtained from an existing database of tRNA families (Sprinzll1996) to construct an initial SCFG for each processor. Appendix A shows the trust multiple alignment for constructing an initial SCFG.

2. Parsing the Training Data in Parallel;

We distribute the training data into different processors and parse them in parallel. Appendix B presents part of the training data.

3. Obtain a big multiple alignment for all training data;

We gather all of the parsed/aligned sequences from all processors to a root processor and then broadcast them from the root processor to all other processors.

4. Constructing a new SCFG;

We use these parsed/aligned sequences to construct a new SCFG for each processor and go back to the second step if the new SCFG is unstable according to some criteria otherwise the new SCFG is stored and the training process stopped.

4.1.2 Test results for Parallel Single-SCFG Training

We have used different number of processors and different number of training data to construct a SCFG. The test results indicate that our training method is more efficient compared to the training process in COVE (Eddy1994).

The time complexity of constructing a CM in COVE is $O(a.L^3.M^3)$, L is the average length of the parsing sequence and M is the number of the nonterminals/states, "a" is a constant factor which is proportion to the number of training sequences.

In our Parallel Single-SCFG Training model, "a" proportion to $1/N$, where N is the number of the processors.

As the number of the processors increased, the total constructing time decreased significantly. A typical example is that constructing a CM using 1400 training data needs 4399 seconds in COVE while it only spends 31 seconds to construct a SCFG using 70 CPUs with the same training data on SHARC-NET.

Table 4.1 summarizes the time complexity of constructing a SCFG using different number of processors and training data on Davinci. It also presents the test results of COVE (second row in the table).

Figure 4.1 also depicts the time complexity of constructing a SCFG using different number of processors and different number of training data on Davinci.

Figure 4.2 depicts the time complexity of using COVE to construct a CM with different number of training data on Davinci.

Table 4.2 summarizes the time complexity of constructing a SCFG using different number of processors with 1400 training data on SHARC-NET.

Figure 4.3 also depicts the time complexity of constructing a SCFG using different number of processors with 1400 training data on SHARC-NET.

CPU Numbers	Time Consumed for different training data													
	100	200	300	400	500	600	700	800	900	1000	1100	1200	1300	1400
2	126	250	239	306	414	872	619	684	1748	1472	924	2518	3658	2492
3	103	171	161	204	283	585	412	446	943	752	609	1648	1009	1632
4	79	125	120	153	207	433	306	341	699	572	444	1066	779	1273
5	63	102	98	122	174	203	248	274	568	443	379	705	608	651
6	54	87	82	102	141	168	204	227	478	469	302	592	500	545
7	51	75	73	92	122	259	179	198	431	327	260	531	438	603
8	46	67	64	79	108	130	160	177	451	303	730	452	418	509
9	45	61	58	74	97	119	142	160	346	270	213	319	350	375
10	43	56	52	69	91	109	130	203	308	239	255	366	317	507
11	39	51	49	61	80	171	119	137	290	222	175	423	694	472
12	24	47	44	56	74	154	156	172	347	276	214	313	390	288
Using COVE	270	526	498	654	882	1850	1304	1418	2365	2397	1904	3894	2365	4399

The table 4.1: the time complexity of constructing a SCFG using different number of processors and different number of training data on Davinci

First column gives the number of processors involved to construct a SCFG; the second row gives the different number of training data used to construct a SCFG; the last row gives the time consuming of using different number of training data with COVE. Other entries give the time consuming of constructing a SCFG using different number of processors and training data on Davinci.

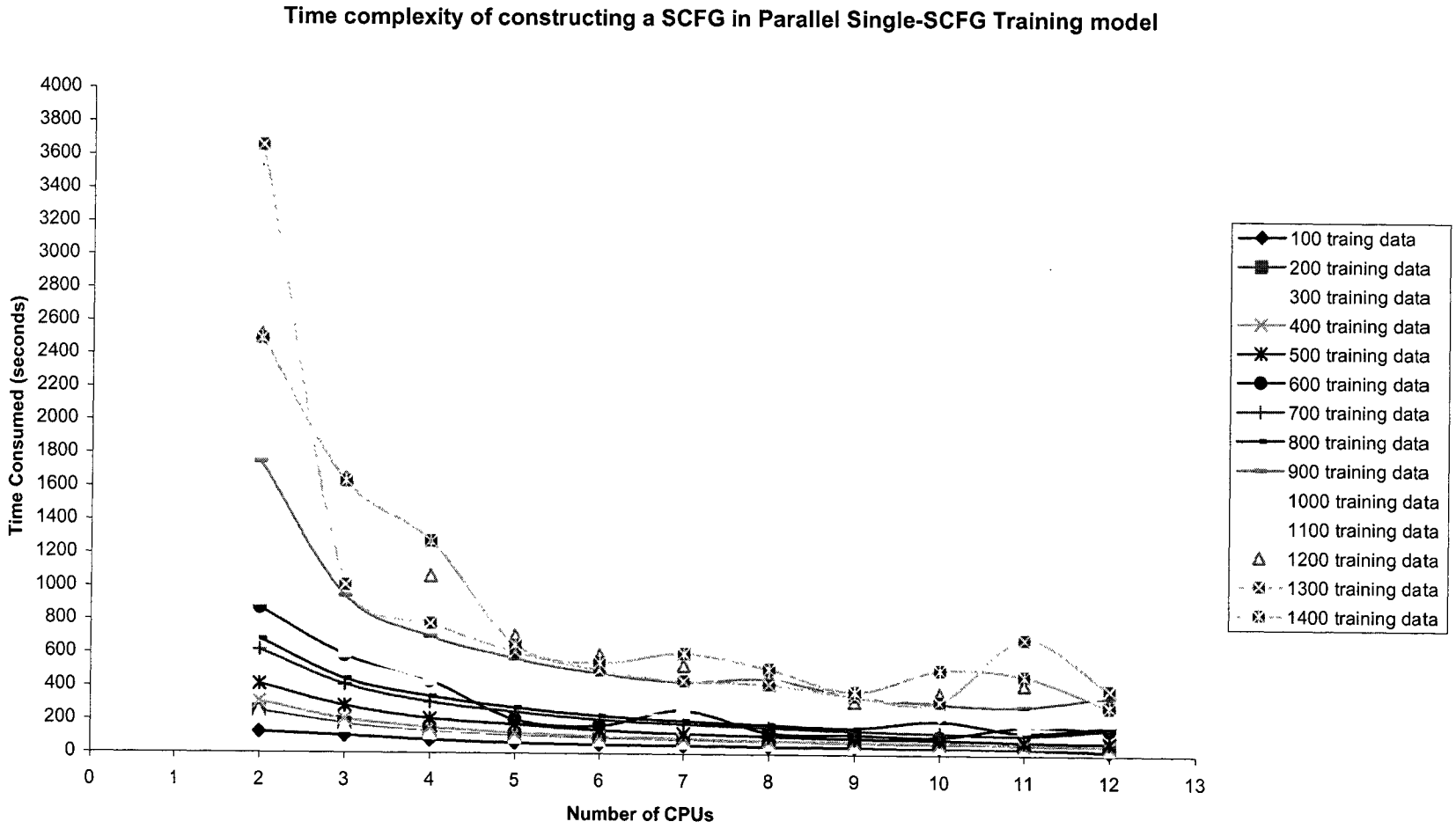

Figure 4.1: Time complexity of using different number of CPUs and training data to construct a SCFG in the Parallel Single-SCFG Training model

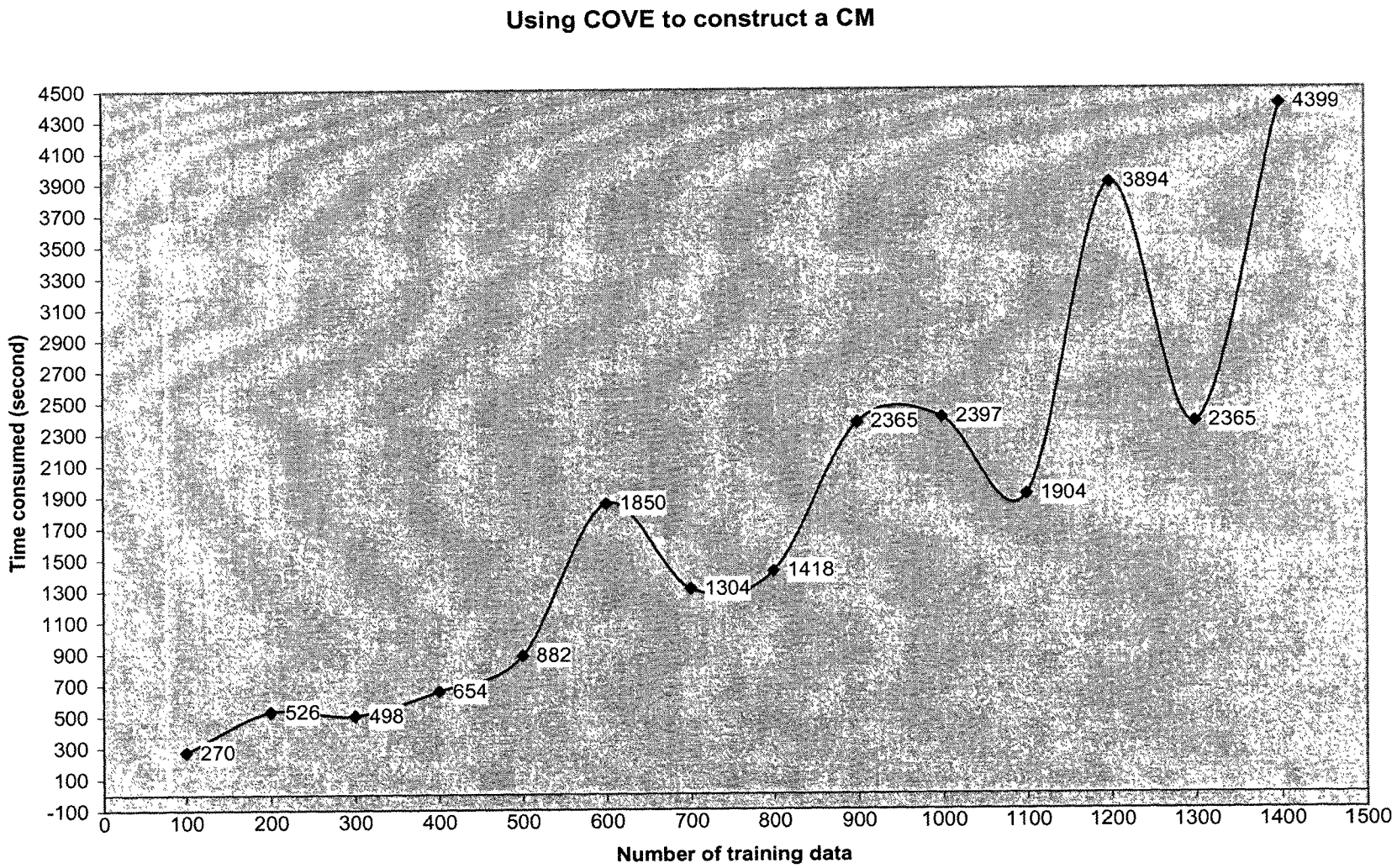

Figure 4.2: The time complexity of using COVE with different number of training data to construct a CM on Davinci

CPU Numbers	Iterations	Time Consumed
10	4	212
20	5	153
30	5	105
40	4	72
50	5	78
60	4	70
70	4	31

Table 4.2: The time complexity of constructing a SCFG using different number of processors and 1400 training data in the Parallel Single-SCFG Training model on SHARCNET; the second column gives the different number of iterations to train a SCFG until it is stable.

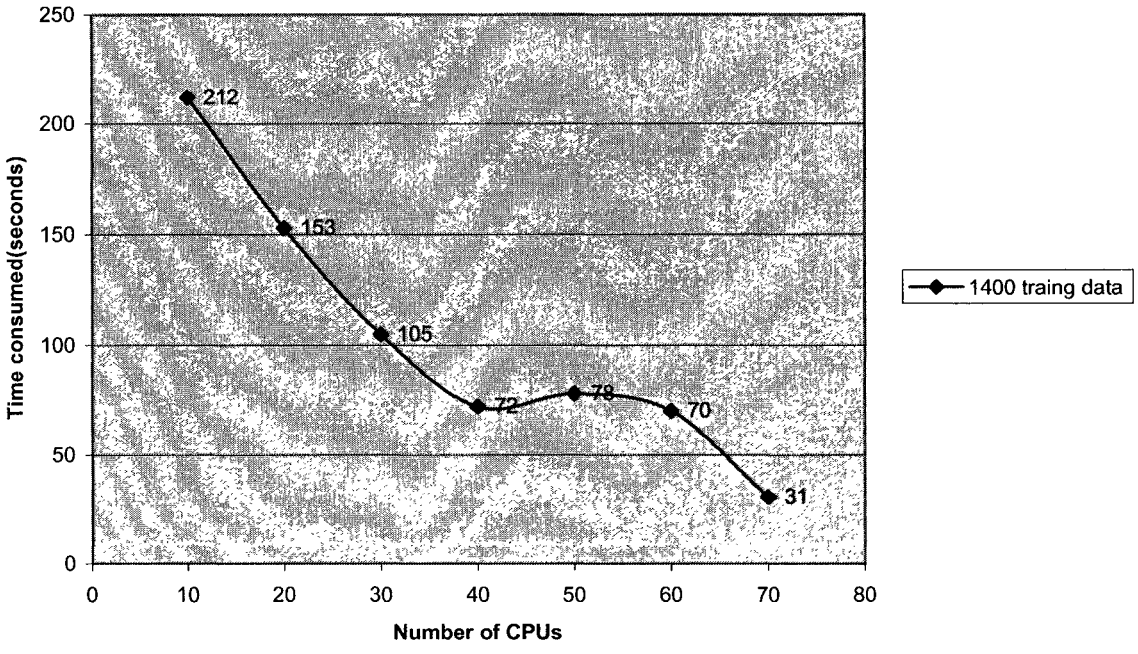

Figure 4.3: The time complexity of the Parallel Single-SCFG Training model on SHARCNET

4.1.3 tRNA secondary structure prediction and multiple alignment

After constructing a SCFG, we use it to produce a multiple alignment. As expected, the test results are exactly same as in COVE.

Appendix C presents a multiple alignment and the secondary structure annotation of five test sequences produced by our SCFG and COVE.

The test results indicate that our SCFG has exactly same accuracy as in COVE.

4.2 Parallel Multi-SCFG Training model

4.2.1 SCFG construction

In this Parallel Multi-SCFG Training method, we apply six steps to construct a SCFG.

The following summarizes these steps:

1. Constructing an initial Sub-SCFG for each processor

We use a trust multiple alignment to construct an initial Sub-SCFG for each processor. This step is exactly same as what we do in the Parallel Single-SCFG Training model. Appendix A gives the trust multiple alignment for constructing an initial-SCFG.

2. Sequentially training the Sub-SCFG

We distribute the training data into different processors and use them to train the initial Sub-SCFG for each processor until it remains stable according to some convergence criteria.

Appendix B presents part of the unaligned data for training a Sub-SCFG.

3. Obtaining a multiple alignment and consensus structure from each processor when each Sub-SCFG is stable.

We obtain a multiple alignment and a consensus structure for the training data for each processor.

Appendix D presents an example of ten multiple alignments and their related consensus structure sequences after the corresponding Sub-SCFGs stable.

4. Combining the consensus structures into a profile

We combine the consensus structures together from all processors into a multiple alignment of consensus structure using the Center-Star like algorithm. Appendix E presents

the initial consensus secondary structure sequences from ten Sub-SCFGs and the multiple alignment of their aligned sequences.

5. Obtaining a big multiple alignment to construct an initial combined-SCFG

We use the multiple alignment of consensus structure to combine a big multiple alignment of all training data and broadcast the big multiple alignment to all processors. After that we construct an initial combined-SCFG for each processor using the big multiple alignment. Appendix F presents the big multiple alignments after aligning ten multiple alignment obtained from ten Sub-SCFGs.

6. Training the combined-SCFG in parallel.

We train the combined-SCFG in parallel until it is stable. This step combines the whole process of the previous Parallel Single-SCFG Training model.

4.2.2 Test results for Parallel Multi-SCFG Training model 1

We have tested this model with different number of processors and different number of training data to construct a SCFG.

The table 4.3 summarizes the time complexity of constructing a SCFG using different number of processors on Davinci. It also presents the test results of COVE (second row in the table).

Compared with COVE and Parallel Single-SCFG Training model 1, we can conclude that, as the number of the processors increased, the total constructing time decreased significantly. The Parallel Multi-SCFG Training model 1 is efficient than COVE but less efficient than the Parallel Single-SCFG Training model.

A typical example is that constructing a CM using 1400 training data needs 4399 seconds in COVE while it only spends 31 seconds to construct a SCFG using 70 CPUs in

the Parallel Single-SCFG Training model on SHARCNET and it needs 870 seconds to construct a SCFG using 12 CPUs in the Parallel Multi-SCFG Training model 1.

Number of processors	Number of training data	Total time/ seconds
2	1400	5183
3	1400	3094
4	1400	1793
5	1400	1728
6	1400	1418
7	1400	1395
8	1400	1294
9	1400	973
10	1400	910
11	1400	935
12	1400	870

Table 4.3: the time complexity of Parallel Multi-SCFG Training model

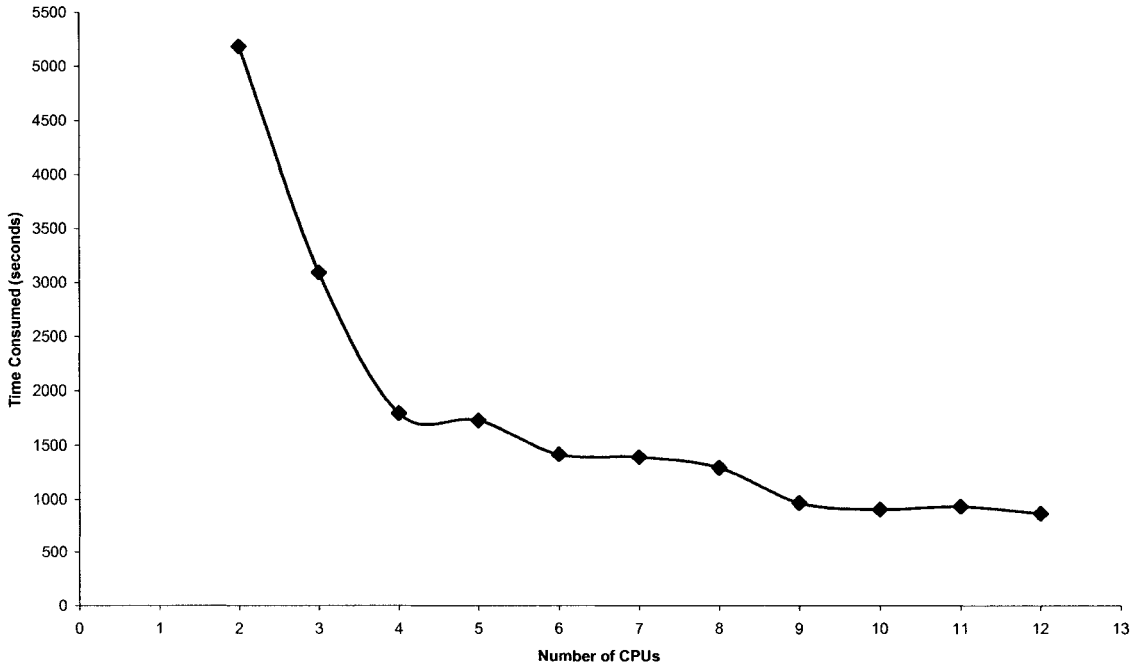

Figure 4.4: The time complexity of Parallel Multi-SCFG Training model 1

4.2.3 tRNA secondary structure prediction and multiple alignment

In order to test the accuracy of the SCFGs, 100 test data are randomly selected from the total 1400 data and tested on those SCFGs, which were constructed with different number of training data using 10 processors on Davinci.

Appendix G presents a table with the alignment/parsing scores for the test data (we can use the scores to detect sequences which do not fit the model if they get negative scores).

Appendix H presents the alignment scores for the same 100 test data tested on the CMs constructed with COVE using the same training data.

The test results show that some of our test results are more accurate than that of the CM and some of our test results are less accurate than that of the CM.

An example of such results is that there are total 67 out of 100 test data have higher alignment scores (tested on the SCFG constructed in the Parallel Multi-SCFG Training model using 1000 training data) while there are total 33 out of 100 test data have higher scores (tested on the CM constructed in COVE using the same 1000 training data).

The test results also indicate even with small number of training data, such as 200 training data, the SCFGs constructed in the Parallel Multi-SCFG Training model 1 can also produce accurate outputs.

5 FUTURE WORK

Our methods have some limitations.

First, not all RNA structures can be captured by an SCFG because the associated parse trees cannot capture tertiary interactions such as pseudoknots and non-pairwise interactions which are the major forces to form a tertiary structure from the secondary structure.

Second, our methods require a reliable and portable computing environment including both of the hardware and software such as SHARC-NET.

Third, since our methods have been tested on short RNA sequences (tRNA), we need to do more tests to model larger and more challenging sequences, such as rRNA and proteins (it should be indicated that using SCFG to predict the secondary structure and/or other higher structures of proteins is more complicated due to the terminal alphabet set is more larger, there are 20 major amino acids exist in living world).

Fourth, we did not do the grammatical inference in our methods because optimal grammatical inference is a NP-complete problem; our methods cannot solve such problem.

Finally, we did not implement the Parallel Multi-SCFG Training model 2 even though the major algorithms are same as in the Parallel Multi-SCFG Training model 1.

Such limitations lead to some directions for future work, summarized as follow.

- **RNA secondary structure prediction with kseudonots**
- **Protein secondary structure and other larger RNA secondary structure prediction**
- **Parallel Multi-SCFG Training model 2**
- **Grammatical inference**

Since Our parallel computing methods are more efficient than the sequential methods and they offer an potential method to develop hybrid SCFG (in the Parallel Multi-SCFG Training model 1and model2), their advantages also lead to some obvious directions for future work, summarized as follow.

- **Parallel computing for Free-Energy Minimization**
- **Parallel computing for multiple sequence alignment**
- **Distributed computing**
- **Parallel computing for dynamic programming algorithms**
- **Parallel database searching and data mining**
- **Developing hybrid SCFG/NN architectures, where an NN is used to compute the parameters of a SCFG to mix different SCFGs .**

6 CONCLUSION

In this thesis, the parallel computing algorithms are applied to train a stochastic context-free grammar for tRNA secondary structure prediction.

Two different parallel methods are implemented in this thesis. One is the Parallel Single-SCFG Training model and the other is the Parallel Multi-SCFG Training model.

In the Parallel Single-SCFG Training mode, the training data is distributed into different processors and parsed in parallel, after that the parallel aligned training data are collected to construct a new SCFG.

In the Parallel Multi-SCFG Training model, we applied a novel algorithm, Center-Star-like embedded in pairwise dynamic programming algorithm, to combine Sub-SCFG together to form a combined-SCFG.

Both of the Parallel Single-SCFG model and Parallel Multi-SCFG Training model are implemented using C language and Message Passing Interface on Davinci.

In order to check the scalability of these methods, the Parallel Single-SCFG model is also implemented on SHARC-NET.

The test results of the Parallel Single-SCFG Training model demonstrate that the methods in this thesis to train a SCFG are more efficient than the sequential training methods in term of time complexity as the number of processors are increased.

The accuracy of the SCFG constructed from the Parallel Single-SCFG Training model is exactly same as the results of that of COVE.

The test results of the Parallel Multi-SCFG Training model also demonstrate that the process of training a SCFG is more efficient than the process of COVE. The accuracy of the

SCFG constructed with the Parallel Multi-SCFG Training model is not less than the test results of COVE.

Once a SCFG was trained, it can be used to produce a multiple alignment of all the training sequences; it can be used to search the database for other sequences that are members of the tRNA family; it can be used to predict the secondary structures of a new tRNA sequence.

Although these methods are tested with tRNA families, it is obvious that these methods could be applied to large RNA families.

BIBLIOGRAGHY

Akmaev V.R., Kelley S.T., and Stormo G.D., **Phylogenetically enhanced statistical tools for RNA structure prediction.** *Bioinformatics*, 16(6): 501-512, 2000.

Alberts B., Bray D., Lewis J., Raff M., Roberts K., and Watson J.D., **Molecular biology of the cell.** *New York : Garland Pub.,* 1994.

Antao V.P., and Tinoco I., **Thermodynamic parameters for loop formation in RNA and DNA hairpin tetraloops.** *Nucleic Acids Research*, 20(4): 819-824, 1992.

Batey R., Rambo R.P., and Doudna J.A., **Tertiary motifs in RNA structure and folding.** *Angew. Chem. Int. Ed.*, 38: 2326, 1999.

Batenburg F.H.D.V., Gultyaev A.P., and Pleij C.W.A., **PseudoBase: structural information on RNA pseudoknots.** *Nucleic Acids Research*, 29: 194-195, 2001.

Baxevanis A.D., **The Molecular Biology Database Collection: 2003 update.** *Nucleic Acids Research*, 31: 1-12, 2003.

Benson D.A., Boguski M.S., Lipman D.J., Ostell J., and Francis B.F., **GenBank.** *Nucleic Acids Research*, 30: 17-20, 2002.

Bockhorst J., and Craven M., **Refining the Structure of a Stochastic Context-Free Grammar.** *Proceedings of the 17th International Joint Conference on Artificial Intelligence* (IJCAI-2001), 2001.

Bokhari S.H, and Sauer J.R., **Sequence Alignment on the Cray MTA-2.** *Second IEEE International Workshop on High Performance Computational Biology Nice, France*, April 22, 2003.

Borer P.N., Dengler B., Tinoco I.J., and Uhlenbeck O.C., **Stability of ribonucleic acid double-stranded helices.** *Journal of Molecular Evolution*, 1974.

Bourdeau V., Ferbeyre G., Pageau M., Paquin B., and Cedergren R., **The distribution of RNA motifs in natural sequences.** *Nucleic Acids Research*, 27(22): 4457-4467, 1999.

Bouthinon D., and Soldano H., **A new method to predict the consensus secondary structure of a set of unaligned RNA sequences**. *Bioinformatics*, 15(10): 785-798, 1999.

Brendel V., and Busse H. G., **Genome structure described by formal languages**. *Nucleic Acids Research*, 12: 2561-2568, 1984.

Brion P., and Westhof e., **Hierarchy and dynamics of RNA folding**. *Annu. Rev. Biophys. Biomol. Struct.* 26: 113, 1997.

Brown M., and Wilson C., **RNA Pseudoknot Modeling Using Intersections of Stochastic Context Free Grammars with Applications to Database Search**. Michael Brown, *Computer and Information Sciences*, Charles Wilson, Department of Biology, University of California, Santa Cruz, CA 95064, USA, 1995.

Carrillo H., and Lipman D., **The Multiple Sequence Alignment Problem in Biology**. *SIAM J Appl Math,* 48(5): 1073-1082, 1988.

Cech T.R., **RNA as an enzyme**. *Scientific American*, 1986.

Cech T.R., **The chemistry of self-splicing RNA and RNA enzymes.** *Science,* 1987.

Chen J.H., Le S.Y., Shapiro B., Currey K.M., and Maizel J., **A computational procedure for assessing the significance of RNA secondary structure**. *Comput. Applic. Biosci.* 6: 7-18, 1990.

Corpet, F. and Michot, B., **RNAlign: alignment of RNA sequences using both primary and secondary structures**. *Comp. Appl. Biosci.* 10, 389-399, 1994.

Durbin R., Eddy S., Krogh A., and Mitchison G., **Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids**. *Cambridge University Press*, 1998.

Dam, E.t., Pleij, K. & Draper, D., **Structural and functional aspects of RNA pseudoknots**. *Biochemistry*, 31, 11665-11676, 1992.

Dirheimer G., Keith G., Dumas P., and Westhof E., **RNA: Structure, Biosynthesis and Function chapter Primary, Secondary, and Tertiary Structures of tRNAs.** *American Society for Microbiology, Washington, DC*, 1995.

Earley J., **An efficient context-free parsing algorithm.** *Communications of the ACM,* 6(8): 451-455, 1970.

Eddy S.R., and Durbin R., **RNA sequence using covariance models.** *Nucleic Acids Research*, 22: 2079-2088, 1994.

Eddy S.R., **A memory-efficient dynamic programming algorithm for optimal alignment of a sequence to an RNA secondary structure.** BMC Bioinformatics. 2, 3(1): 18, 2002.

[Fitch1983] Fitch W.M., and Smith T.F., **Optimal sequence alignments.** *Proceedings of the National Academy of Science of the USA*, 80: 1382-1386, 1983.

Fontana W., Konings D.A.M., Stadler P.F. and Schuster P., **Statistics of RNA secondary structures.** *Biopolymers*, 33: 1389-1404, 1993.

Ganot P., Bortolin M.L., and Kiss T., **Site-specific pseudouridine formation in preribosomal RNA is guided by small nucleolar RNAs.** *Cell*, 1997.

Garian R., **Prediction of quaternary structure from primary structure.** *Bioinformatics,* 17: 551-556, 2001.

Gautheret D., Major F., and Cedergren R., **Pattern searching alignment with RNA primary and secondary structures: an effective descriptor for tRNA.** *CABIOS,* 6(4): 325-331, 1990.

Giege, R., Puglisi, J. D. & Florentz, C., **tRNA structure and aminoacylation efficiency.** *Prog. Nucl. Acid Res. Mol. Biol.* 45: 129-206, 1993.

Grate L., Herbster M., Hughey R., Mian I.S., Noller H., and Haussler D., **RNA Modeling Using Gibbs Sampling and Stochastic Context Free Grammars.** *Proceedings, 2nd International Conference on Intelligent Systems for Molecular Biology*, 235: 1501-1531, 1994.

Graham S.L, Harrison M.A., and Ruzzo W.L., **An improved context-free recognizer.** *ACM Transactions on Programming Languages and Systems,* 2(3): 415-462, 1980.

Graber G.H., McAllister G.D., and Smith T.F., **Probabilistic prediction of Saccharomyces cerevisiae mRNA 3'-processing sites.** *Nucleic Acids Research*, 30: 1851-1858, 2002.

Gribskov M., Luethy R., and Eisenberg D., **Profile Analysis.** *In Methods in Enzymology,* 183: 146-159, 1989.

Gropp W., Lederman H.S., Lumsdaine A., Lusk E., Nitzberg B., Saphir W., and Snir M., **MPI - The Complete Reference, Volume 2, The MPI Extensions.** The MIT Press 1998.

Gruner W., Giegerich R., Strothmann D., Reidys C., Weber J., Hofacker I.L., Schuster P., and Stadler P.F., **Analysis of RNA sequence structure maps by exhaustive enumeration.** *Monatshefte fur Chemie*, 127(4): 355-374, 1996.

Gultyaev A.P., Vanbatenburg F.H.D., and Pleij C.W.A., **The computer-simulation of RNA folding pathways using a genetic algorithm.** *Journal of Molecular Biology*, 250(1): 37-51, 1995.

Gutell R., Power A., Hertz G., Putz E., and Stromo G., **Identifying constraints on the higher-order structure of RNA: continued development and application of comparative sequence analysis methods.** *Nucleic Acids Research,* 20: 5785-5795, 1992.

Han K., and Kim H.J., **Prediction of common folding structures of homologous RNAs.** *Nucleic Acids Research*, 21: 1251-1257, 1993.

Herschlag D., **RNA chaperones and the RNA folding problem.** *J. Biol. Chem.,* 270: 20871, 1995.

Hofacker I.L., Fekete M., Flamm C., Huynen M.A., Rauscher S., Stolorz P.E., and Stadler P.F., **Automatic detection of conserved RNA structure elements in complete RNA virus genomes.** *Nucleic Acids Research*, 26(16): 3825-3836, 1998.

Hofacker I.L., Schuster P., and Stadler P.F., **Combinatorics of RNA secondary structures.** *Discrete Appl. Math.*, 88(1-3): 207-237, 1998.

Hofacker I.L., Fontana W., Bonhoeffer S., and Stadler P.F., **Fast folding and comparison of RNA secondary structures.** *Monatshefte fur Chemie*, 125:167-188, 1994.

Hunt J.W., and Szymanski T.G., **A fast algorithm for computing longest common subsequences**. *Communications of the ACM*, 20: 350-353, 1977.

James B.D., Olsen G.J., and Pace N.R., **Phylogenetic comparative analysis of RNA secondary structure**. *Meth. Enzymol.*, 180: 227-239, 1989.

Jaeger J.A., Turner D.H., and Zuker M., **Predicting optimal and suboptimal secondary structure for RNA**. *Methods Enzymol.* 183: 281-306, 1990.

Karlin S., Morris M., Ghandour G., and Leung M.Y., **Efficient algorithms for molecular sequence analysis**. *Proceedings of the National Academy of Science of the USA*, 85: 841-845, 1988.

Kim S.H., Suddath G.J., Quigley G.J., McPherson A., Sussman J.L., Wang A.H.J., Seeman N.C., and Rich A., **Three-dimensional tertiary structure of yeast phenylalanine transfer RNA**. *Science*, 185: 435-439, 1974.

Konings D.A., and Hogeweg P., **Pattern analysis of RNA secondary structure: similarity and consensus of minimal-energy folding**. *Journal of Molecular Biology*. 207: 597-614, 1989.

Krogh A., Brown M., Mian S., Sjölander K., and Haussler D., **Hidden Markov Models in Computational Biology: Applications to Protein Modeling**, *Journal of Molecular Biology*, 4; 235(5): 1501-1531, 1994.

Knudsen B., and Hein J., **RNA Secondary structure prediction using stochastic context-free grammars and evolutionary history**. *Bioinformatics,* 15(6): 446-454, 1999.

Kumar V., Grama A., Gupta A., and Karypis G., **Introduction to Parallel Computing, Design and Analysis of Algorithms**. *The Benjamin/Cummings Publishing Company, Inc.* 1994

[Lari1990] Lari K., and Young S.J., **The estimation of stochastic context-free grammars using the Inside-Outside algorithm**. *Computer Speech and Language*, 4: 35-56, 1990.

Lawrence C., and Reilly A., **An Expectation Maximization (EM) Algorithm for the Identification and Characterization of Common Sites in Unaligned Biopolymer Sequences**. *Proteins,* 7(1): 41-51, 1990.

Lawrence C., Altschul S.F., Boguski M.S., Liu J.S., Neuwald A.F., Wootton J.C., **Detecting Subtle Sequence Signals: A Gibbs Sampling Strategy for Multiple Alignment**. *Science,* 262: 208-214, 1993.

Leslie G., **Automatic RNA Secondary Structure Determination with Stochastic Context-Free Grammars**. *The Third International Conference on Intelligent Systems for Molecular Biology* July 16-19, 1995.

Le S.Y., Chen J.H., Currey K.M., and Maizel J., **A program for predicting significant RNA secondary structures**. *Comput. Applic. Biosci.*, 1988.

Le S.Y., Owens J., Nussinov R., Chen J.H., Shapiro B., and Maizel J.V, **RNA secondary structures: comparison and determination of frequently recurring substructures by consensus**. *Comput. Appl. Biosci.* 5: 205-210, 1989.

Le S.Y., and Zuker M., **Predicting Common Folding of Homologous RNAs**. *J. Biomolecular Structure and Dynamics,* 8: 1027-1044, 1991.

Lowe T.M., and Eddy S.R., **tRNAscan-SE: A program for improved detection of transfer RNA genes in genomic sequence**. *Nucleic Acids Research*, 25(5): 955-964, Mar 1997.

Lowe T.M., and Eddy S.R., **A computational screen for methylation guide snoRNAs in yeast**. *Science*, 283(5405): 1168-1171, 1999.

Lyngsø R.B., and Pedersen C.N.S., **Pseudoknots in RNA secondary structures**. *In Proceedings of the 4th Annual International Conference on Computational Molecular Biology (RECOMB),* pages 201-209, 2000.

Macke T.J., Ecker D.J., Gutell R.R., Gautheret D., Case D.A., Sampath R., **RNAMotif, an RNA secondary structure definition and search algorithm**. *Nucleic Acids Research,* 15; 29(22): 4724-35, 2001.

Maden B., **The numerous modified nucleotides in eukaryotic ribosomal RNA**. *Progr Nucleic Acid Res Mol Biol*, 1990.

Marvel C.C., **A program for the identification of tRNA-like structures in DNA sequence data**. *Nucleic Acids Research,* 14: 431-435, 1986.

Mathews D.H., Sabina J., Zuker M., and Turner D.H., **Expanded sequence dependence of thermodynamic parameters improves prediction of RNA**. *Journal of Molecular Biology*, 288(5): 911-940, 1999.

Moll R.N., Arbib M.A., and Koufry A.J., **An introduction to formal language theory**. *Springer-Verlag, New York*, 1988.

Moulton V., Zuker M., Steel M., Pointon R., and Penny D., **Metrics on RNA secondary structures**. *Journal of Computational Biology*, 7(1-2): 277-292, 2000.

Nakaya A., Yamamoto K., and Yonezawa A., **RNA secondary structure prediction using highly parallel computers**. *Compt. Appl. Biosci.*, 11: 685-692, 1995.

Noller, H.F., Kop, J., Wheaton, V., Brosius J., Gutell R.R., Kopylov A.M., Dohme F., Herr W., Stahl D.A., Gupta R. and Woese C.R., **Secondary structure model for 23S ribosomal RNA**. *Nucleic Acids Research,* 9: 6167-6189, 1981.

Noller H.F., and Woese C.R., **Secondary structure model for 16S ribosomal RNA.** *Science*, 212: 403-411, 1981.

Nussinov R., Pieczenik G., Griggs J.R., and Kleitman D.J., **Algorithms for loop matchings**. *SIAM Journal on Applied Mathematics*, 35: 68-82, 1978.

Nussinov R., and Jacobson A.B., **Fast algorithm for predicting the secondary structure of single-stranded RNA**. *Proceedings of the National Academy of Sciences of the USA,* 77: 6309-6313, 1980.

Pace N., Smith D., Olsen G., and James B., **Phylogenetic comparative analysis and the secondary structure of ribonuclease**. *Gene*, 1989.

Papanicolaou C., Gouy M., and Ninio J., **An energy model that predicts the correct folding of both the tRNA and the 5S RNA molecules**. *Nucleic Acids Research*, 1984

Pleij C.W.A., Rietveld K., and Bosch L., **A new principle of RNA folding based on pseudoknotting**. *Nucleic Acids Research,* 13: 1717-1731, 1985.

Poole A.M., Jeffares D.C., and Penny D., **The path from the RNA world**. *Journal of Molecular Evolution*, 1998.

Rivas E., and Eddy S.R., **A dynamic programming algorithm for RNA structure prediction including pseudoknots**. *Journal of Molecular Biology*, 285(5): 2053-2068, 1999.

Rivas E., and Eddy S.R., **The language of RNA: a formal grammar that includes pseudoknots**. *Bioinformatics,* 16: 334-340, 2000.

Rivas E., and Eddy S.R., **Secondary structure alone is generally not statistically significant for the detection of noncoding RNAs**. *Bioinformatics*, 16(7): 583-605, Jul 2000.

Sakakibara Y., Brown M., Hughey R., Mian I.S., Sjolander K., Underwood R.C., and Haussler D., **Stochastic context-free grammars for tRNA modeling**. *Nucleic Acids Research*, 22: 5112-5120, 1994.

Sankff D., **Matching sequences under deletion/insertion constraints**. *In Proceedings of the National Academy of Science of the USA*, 69: 4-6, 1972.

Sankoff D., **Simultaneous solution of the RNA folding, alignment, and protosequence problems**. *SIAM J. Appl. Math.*, 45: 810–825, 1985.

Sankoff D., Morin A., and Cedergren R., **The evolution of 5S RNA secondary structures**. *Canadian Journal of Biochemistry*, 56: 440-443, 1978.

Schmidt B., Schorder H., and Schimmler M., **Massively Parallel Solutions for Molecular Sequence Analysis.** *First IEEE International Workshop on High Performance Computational Biology Fort Lauderdale, FL*, April 15, 2002.

Schmitt W.R., and Waterman M.S., **Linear trees and RNA secondary structure.** *Discr. Appl. Math.*, 12: 412-427, 1994.

Schmitz M., and Steger G., **Base-pair probability profiles of RNA secondary structures.** *Comp. Appl. Biosci.* 8: 389-399, 1992.

Schultes E.A., Hraber P.T., and LaBean T.H., **Estimating the contributions of selection and self-organization in RNA secondary structure.** *Journal of Molecular Evolution*, 49(1): 76-83, 1999.

Searls D.B., **Linguistics approaches to biological sequences.** *CABIOS,* 13: 333-344, 1997.

Seffens W. and D. Digby. **mRNAs have greater negative folding free energies than shuffled or codon choice randomized sequences.** *Nucleic Acids Research*, 27(7): 1578-1584, 1999.

Shapiro B.A., and Zhang K., **Comparing multiple RNA secondary structures using tree comparisons.** *Comput. Appl. Biosci.*, 6: 309-318, 1990.

Snir M., Otto S., Lederman H.S., Walker D., and Dongarra J., **MPI - The Complete Reference, Volume 1, The MPI Core, second edition.** *MIT Press* 1998.

Sprinzl M., Steegborn C., Hubel F., and Steinberg S., **Compilation of tRNA sequences and sequences of tRNA genes.** *Nucleic Acids Research,* 24: 68-72, 1996.

Staden R., **A computer program to search for tRNA genes.** *Nucleic Acids Research*, 8: 817-825, 1980.

Stephan W., Parsch J, Braverman JM., **Comparative sequence analysis and patterns of covariation in RNA secondary structures.** *Genetics*, 154(2): 909-921, 2000.

Stolcke A., **An Efficient Probabilistic Context-Free Parsing Algorithm that Computes Prefix Probabilities.** *Computational Linguistics*, 21(2): 165-201, 1995.

Sussman, J.L. et al., **Crystal Structure of Yeast Phenylalanine Transfer RNA I. Crystallographic Refinement**. *Journal of Molecular Biology*, 123: 607, 1978.

Tinoco J.I., Uhlenbeck O.C., and Levine M.D., **Estimation of Secondary Structure in Ribonucleic Acids**. *Nature,* 230: 363-367, 1971.

Tuerk C., and Gold L., **Systematic evolution of ligands by exponential enrichment: RNA ligands to bacteriophage T4 DNA polymerase**. *Science*, 1990.

Turner D.H., Sugimoto N., and Freier S.M., **RNA structure prediction**. *Ann. Rev. Biophys. and Biophys. Chem.*, 17: 167-192, 1988.

Voet D., and Voet J.G., **Biochemistry**. *J. Wiley and Sons, Inc.,* 1995.

Voet D., Voet J., and Pratt C.W., **Fundamentals of Biochemistry**. *J. Wiley and Sons, Inc.,* 1999.

Walter A.E., Turner D.H., Kim J., Lyttle M.H., Muller P., Mathews D.H., and Zuker M., **Coaxial stacking of helices enhances binding of oligoribo-nucleotides and improves predictions of RNA folding**. *Proc. Natl Acad. Sci. USA*, 91: 9218-9222, 1994.

Waterman M.S., and Smith T.F., **RNA secondary structure: A complete mathematical analysis**. *Math. Biosci.* 42: 257-266, 1978.

Waterman M.S., and Smith T.F., **Rapid dynamic programming algorithms for RNA secondary structure**. *Adv. Appl. Math.*, 7: 455-464, 1986.

Waterman M.S., **Introduction to Computational Biology: Maps, sequences and genomes**. *Chapman & Hall,* 1995.

Williams A.L., and Tinoco I.J., **A dynamic programming algorithm for finding alternative RNA secondary structures**. *Nucleic Acids Research,* 14: 299-315, 1986.

Winker S., Overbeek R., Woese C., Olsen G., and Pfluger N., **Structure detection through automated covariance search**. *CABIOS*, 6: 365-371, 1990.

Woese and Pace N., **The RNA World, chapter Probing RNA structure, function, and history by comparative analysis**. *Cold Spring Harbor Laboratory Press, Cold Spring* Harbor, NY, 1993.

Woese C.R., **Origin of life**. *Scientific Publications Dept., Carolina Biological Supply Co.,* 1985.

Woese C.R, Gutell R.R., Gupta R., and Noller H.F., **Detailed analysis of the higher-order structure of the 16S-like ribosomal ribonucleic acids**. *Microbiology Review*, 47: 621-669, 1983.

Woodson S.A., **Recent insights on RNA folding mechanisms from catalytic RNA**. *Cell. Mol. Life Sci.*, 57: 796, 2000.

Wuchty S., Fontana W., Hofacker I.L., and Schuster P., **Complete suboptimal folding of RNA and the stability of secondary structures**. *Biopolymers*, 49(2): 145-165, 1999.

Wyatt J.R., Puglisi J.D. and Tinoco, I.J., **RNA folding: pseudoknots, loops and bulges**. *Bioessays*, 11: 100-106, 1989.

Xiao X., Dow E., Eberhart R., Miled Z. B., and Oppelt R. J., **Gene Clustering using Self-Organizing Maps and Particle Swarm Optimization.** *Second IEEE International Workshop on High Performance Computational Biology Nice, France,* April 22, 2003.

Yap T.K., Munson P.J., Frieder O., and Martino R.L., **Parallel multiple sequence alignment using speculative computation**. *Proceedings of the 1995 International Conference on Parallel Processing.* August, 1995

Yap T.K., Munson P.J., Frieder O., and Martino R.L., **Parallel Computation in Biological Sequence Analysis**. *IEEE Transactions on Parallel and Distributed Systems.* Vol. 9, No. 3, March 1998.

Yokomori T., and Kobayyashi S., **DNA Evolutionary Linguistics and RNA Structure Modeling: A Computational Approach.** *Proc. of 1st International IEEE Symposium on Intelligence in Neural and Biological Systems,* 1995.

Zhu J., Liu J.S., and Lawrence C.E., **Bayesian Adaptive Sequence Alignment Algorithms.** *Bioinformatics,* 14: 25–39, 1998.

Zuker M., Stiegler P., **Optimal computer folding of large RNA sequences using thermodynamics and auxiliary information.** *Nucleic Acids Research*, 9: 133–148, 1981.

Zuker M., and Sankoff D., **RNA secondary structure and their prediction.** *Bulletin of Mathematical Biology*, 46: 591-621, 1984.

Zuker M., **On finding all suboptimal foldings of an RNA molecule.** *Science,* 244: 48-52, 1989.

Zuker M., and Le S., **Common structures of the 5' non-coding RNA in enteroviruses and rhinoviruses, thermodynamical stability and statistical significance.** *Journal of Molecular Biology*, 216: 729-741, 1990.

Zuker M., **Calculating nucleic acid secondary structure.** *Current Opinion in Structural Biology*, 10(3): 303-310, 2000.

APPENDIX A: The trust multiple alignment for constructing an initial SCFG

```
DK0660     GGGC CC GU A GCUUAGUC UGGU   AGAGCG C CUGAC UUUUAAUCAG GCG                          GU  CGAG GGU UCG AA  UCC CU U CGGGCCC G
DE2520     GCCC CC AU C GUCUAGA  GGCCU AGGACA U CUCCC UUUCACGGAG GCA                          A   CGGG GAU UCG AA  UUC CC C UGGGGGU A
DT5320     GUCU UG AU A GUAUAA   AC    AUUACU C UGGUC UUGUAAACCU GAA                          A   UGAA G A UCU UC  U C UU C UCAAGAC A
DR1181     GCCC AU GU A GCUCAGUA GGAU  AGAGCA C GCGCC UUCUAAGCGU GAG                          GU  CGGA AGU UCG AG  CCU UC U CGUGGGC ACCA
DC2720     GGCG GC AU G GCCAAGU  GGU   AAGGCA G GGGAC UGCAAAUCCU UUA                          C   CCCC AGU UCA AA  UCU GG G UGCCGCC U
DL5720     ACUU UU AA A GGAUAAC  AGC   CAUCCG U UGGUC UUAGGCCCCA AAA                          A   UUUU GGU GCA AC  UCC AA A UAAAAGU A
DC2680     GGCG GC AU G GCCAAGC  GGU   AAGGCA G GGGAC UGCAAAUCCU UUA                          U   CCCC AGU UCA AA  UCU GG G UGCCGCC U
DM3920     ACCU GG GU A GUUUAAA  GGU   AAAACC U UAAUU UCAUACAUUA AAG                          A   UGAG AAU UCG AU  UUU CU C CCCAGGU N
DD4700     UUAA AA GU A GUUAAU   AU    AUAACU U AGAAU UGUCAGUUCU AGA                          U   UCCU U U UAC U    A AG G CUUUUUA U
DY6280     CUCU CG GU A GCCAAGUU GGUUU AAGGCG C AAGAC UGUAAAUCUU GAG                          AU  CGGG CGU UCG AC  UCG CC C CCGGGAG A
DT5080     GCCU UG AA A GCUCAAC  AACU  AGAGCU U UGGUC UUGUAAACCA GGA                          G   AGAG GGU AAA CU   CC CU C UCAAGGC U
DR4000     GCUC UC UU A GCUUAAU  GGUU  AAAGCA U AAUAC UUCUAAUAUU AAU                          AU  UCCA UGU UCA AA  UCA UG G AGAGAGU A
DL2601     GGGG GU AU G GCGAAAUU GGUA  GACGCU G CGGAC UUAAAAUCCG UU  GGC UU  UAA      AG ACCG  UGAG GGU UCA AG  UCC CU C UACCCCC A
DV6161     GGUC GG AU G GUGUAGUC GGUU  AUCACG G UUGCU UUACACGCAA CAG                          GU  CUCG AGU UCG AU  CCU CG G UCGGAUC A
DH4700     AAAA AA GU A GUUUAAUU UA    AGAACG A UAGGU UGUGGGGCUA UUA                          G   CGGU     GUU A      AC C CUUUUUU A
DK4030     GAGA GU AU   UGUUUAAU GGUA  AAACAU U CUGUC UUUUAAGCAG UCU                          A   UAGU GGU UCG AU  UCC AC U UAUUCUC A
DG2601     GCGG GU AU A GUUUAGU  GGU   AAAACC U UAGCC UUCCAAGCUA ACG                          A   UGCG GGU UCG AU  UCC CG C UACCCGC U
DE1140     GGCC UG UU G GUGAAGC  GGUU  AACACA C ACGGU UUUCAUCCGU GGA                          CA  CACG GGU UCG AA  CCC CG U ACAGGCU ACCA
DS3881     GGGA AA GU   UUCCAUG  GGUA  GGGUAA G AUAUU UGCUAAAUAU UG  UGC G   UUU   G    C ACA    UAGA UGU UCG AA  UCA UC U CUUUCCC G
DY5000     GAUU GG GU G GCAGAUA  GG    GAUGCA C UAGAU UGUAAAUCUA GGU                          A   GGAA GGU UCA AG  UCC UU U CCUGGUC A
DI2580     GGGU UA UU A GCUCAGUU GGUU  AGAGCA C ACCCC UGAUAAGGGU GAG                          GU  CCCU GGU UCA AA  UCC AG G AUAACCC A
DX4880     AAAA AG AU A AGCUAA   UU    AAGCUA C UGGGU UCAUACCCCA UUU                          A   UAAA GGU UAU AA  UCC UU U UCUUUUU A
DD6280     UCCG UG AU A GUUUAAU  GGUC  AGAAUG G GCGCU UGUCGCGUGC CAG                          A   UCGG GGU UCA AU  UCC CC G UCGCGGA G
DS2920     GGAG AG AU G GCUGAGU  GGACU AAAGCG G CGGAU UGCUAAUCCG UU  GUA CGA GUU   AA UCG UAC    CGAG GGU UCG AA  UCC CU C UCUUUCC G
DK7680     GCCC GG CU A GCUCAGUC GGU   AGAGCA U GAGAC UCUUAAUCUC AGG                          GU  CGUG GGU UCG AG  CCC CA C GUUGGGC G
DI5120     GGAA AU GU G CCCGAA   AG    UAGGGA U CACUU UGAUAGAGUG AAA                          UA  UAUG GGU UCA AA  CCC CA U CAUCUCC U
DI4080     GAAA CU AU A AUUCAAUU GGUU  AGAAUA G UAUUU UGAUAAGGUA CCA                          A   UAUA GGU UCG AU  UCC UG U UAGUUUC A
DV6280     GGUU UC GU G GUCUAGUC GGUU  AUGGCA U CUGCU UAACACGCAG AAC                          GU  CCCC AGU UCG AU  CCU GG G CGAAAUC A
DA5040     GUGG UU CU A GUUUAGU  GA    AAAACG U UUGCU UUGCAAGCAG AAA                          U   CCUA AGU UAA AU  UCU UA G GAACUAC A
DY6742     CCGA CC UU A GCUCUGUU GGU   AUAGCG G AGGAC UGUAGAUCCU UAG                          GU  CACU GGU UCG AA  UUC GG U AGGUCGG A
DH2600   G GCGG AC GU A GCCAAGU  GGAUU AAGGCA G UGGAU UGUGGAUCCU CUA                          CG  CGCG GGU UCA AU  UCC CG U CGUUCGC C
DH5880     GUAA AU AU A GUUUAA   CC    AAAACA U CAGAU UGUGAAUCUG ACA                          A   CAGA GGC UUA CG  ACC CC U UAUUUAC C
DT5220     ACUC UA AU A GUUUAU   GA    AAAACA U UGGUC UUGUAAACCA AAA                          AC  UGAA GAC UCC AC   CC UU C UUAGAGU A
DI2440     GGGC UA UU A GCUCAGUU GGUU  AGAGCG U UGCUU UGAUAAGGCA AAA                          GU  CGAA AGU UCA AA  UCU UU C AUAGCCC A
DS6241     GGCG CG AU G GCAGAGU  GGUCU AAUGCG U GAGAC UUGAAAUCUC AU  UUC     UUC   G      GAAG  CGUC GGU UCA AA  UCC GG C UCGCGUC G
DM6160     AGCU UC UU A ACUCAGG  GGU   AGAGUG C GAGGC CCAUAACCUC GAG                          GU  CCUA GGA UCG AA  ACC UA G AGAAGCU A
DT5360     GUCU UU GU A GUACAU   CU    AAUAUA C UGGUC UUGUAAACCA GAG                          A   AGGA GAA CAA CU AACC UC C CUAAGAC U
DA0380     GGGC CC AU A GCUCAGU  GGU   AGAGUG C CUCCU UUGCAAGGAG GAU                          GC  CCUG GGU UCG AA  UCC CA G UGGGUCC A
DW4980     AGAG AU UU A AGUUAA   UA    AAACUG A AAACC UUCAAAGUUU UAA                          A   UAAG AGU GGA AC  UCU CU U AGUCUUU A
DR2700     GCGU CC AU U GUCUAAU  GGAU  AGGACA G AGGUC UUCUAAACCU UUG                          G   UAUA GGU UCA AA  UCC UA U UGGACGC U
DH5840     GUAA AU AU A GUUUAA   CC    AAAACA U UAGAU UGUGAAUCUA AUA                          A   UAGG GCC CCA CA  ACC CC U UAUUUAC C
DA4500     GGGG AC GU A GCUCAAUU GGU   AGAGCG U AUGUU UUGCAAGCAU AAA                          GC  UGUC GGU UCA AA  UCC GA U CGUCUCC A
DQ9991     GUCU CU GU G GCGCAAUC GGUU  AGCGCG U UCGGC UGUUAACCGA AAG                          AU  UGGU GGU UCG AG  CCC AC C CAGGGAC G
DP1140     CGGG AA GU G GCUCAGUU UGGU   AGAGCA U UCGGU UUGGGACCGA AGG                          GU  CGCA GGU UCA AA  UCC UG U CUUCCCG ACCA
```

```
DV0860    GGGC CC GU C GUCUAGCC UGGUU  AGGACG C UGCCC UGACGCGGCA GAA                          AU  CCUG GGU UCA AG  UCC CA G CGGGCCC A
DG1500    GCGG AA GU A GUUCAGU  GGU   AGAACA U CACCU UGCCAUGGUG GGG                          GU  CGCG GGU UCG AA  UCC CG U CUUCCGC U
DG2681    GCGG GU AU A GUUUAGU  GGU   AAAACC C UAGCC UUCCAAGCUA ACG                          A   UGCG GGU UCG AU  UCC CG C UACCCGC U
DP1662    CGGC AC GU A GCGCAGCC UGGU  AGCGCA C CGUCA UGGGGUGUCG GGG                          GU  CGGA GGU UCA AA  UCC UC U CGUGCCG ACCA
DF5920    GUUA AU GU A GCUUUAA  AAUU  AAAGCA A GGCAC UGAAAAUGCC UAG                          A   UGAG U G CUC CA    A CU C CAUAAAC A
DN5000    UGGG UU GU A GCCUAAU  GGA   AAGGCA A UUGGC CGUUAACCAG GAG                          AU  AACA AGA UCA AU  ACU UG U CAACUCA G
DG4501    GCGG AU AU A GAUUAAA  GGU   AAAUUA U CUGCC UUCCAAGCAG AGG                          A   UAUG GGU UCG AU  UCC CG U UAUCCGC A
DC4840    GGUC UU AU A GUCAAUA        AUGAUA U CAAAC UGCAAUUUUG AAG                          G   AGUA A   GUU       U UA C UAAGGCU U
DC5160    GGCC UU GG G GUGUC          AACACG U GGGGU UGCAAACCCC AAG                          A   UGCA GUA UAA UA   CC UG C CGGGGCU U
DD1180    GGCC CC AU A GCGAACGU UGGUU  AUCGCG C CUCCC UGUCACGGAG GAG                          AU  CACG GGU UCG AG  UCC CG U UGGGGUC GCCA
DA2520    GGGG GU AU A GCUCAGUU GGU   AGAGCG C UGCCU UUGCAAGGCA GAU                          GU  CAGC GGU UCG AG  UCC GC U UAUCUCC A
DN6060    GCUC GA UU A GCUCAGCU GGUU  AGAGCA U GCGGC UGUUAACCGC AAG                          GU  CGUA GGU UCG AU  CCC UA C AUCGAGC G
DF5970    GUUC AU GU A GCUUAAA  ACC   AAAGCA A GGCAU UGAAAAUGCC UAG                          A   UGAG U A UAU UA    A CU C CAUAAAC A
DD1140    GGCC CC AU A GCGAAGUU GGUU  AUCGCG C CUCCC UGUCACGGAG GAG                          AU  CACG GGU UCG AG  UCC CG U UGGGGUC GCCA
DF7740    GCCG AA AU A GCUCAGUU GGG   AGAGCG U UAGAC UGAAGAUCUA AAG                          GU  CCCC GGU UCA AU  CCC GG G UUUCGGC A
DW5040    AAGA GC UU A AGUUAA   AU    AAACUG A AAGCC UUCAAAGCUU UUU                          A   UAAG AAU GGA AA  CUU CU U AGCUCUU G
DF4700    GAUA CG GU A GCUUAA   UU    AAAGCG U CUCAU UGAAAAUGAG GAA                          GA  UGGU AC  UUU A   GU AC C UUGUGUC A
DP7560    GGCC GA AU G GUCUAGU  GGU   AUGAUU C UCGCU UUGGGUGCGA GAG                          GU  CCCG GGU UCA AU  CCC CG G UUCGGCC C
DW3960    AAGA GU AU A GUUUAAA  GGU   AAAACA G AAAGC UUCAACCUUU AAU                          U   UCUU AGU UCG AG  UCU AA G UGCUCUU G
DA2540    GGGG GU AU A GCUCAGUU GGU   AGAGCG C UGCCU UUGCACGGCA GAU                          GU  CAGG GGU UCG AG  UCC CC U UACCUCC A
DA1540    GGGG CC UU A GCUCAGCU GGG   AGAGCG C CUGCU UUGCACGCAG GAG                          GU  CAGC GGU UCG AU  CCC GC U AGGCUCC A
DQ4440    UGGA GU AU A GCCAAGU  GGU   AAGGCA U CGGUU UUUGGUAUCG GCA                          UG  CAAA GGU UCG AA  UCC UU U UACUCCA G
DL5560    ACUU UU AA A GGAUAAC  AGC   UAUCCA U UGGCC UUAGGAGUCA AAA                          A   UAUU GGU GCA AC  UCC AA A UAAAAGU A
DL2180    GGGG AU GU G GUGGAAUU GGUA  GACGCA A CGGAC UUAAAAUCCG UC  GAU U   GUA   UA   G AUCG   UGAG GGU UCA AG  UCC CU C CGUCCCC A
DX0980    AGCG GG GU A GGCCAGCC AGGU  AGGCCG C GGGGC UCAUANCCCC GAG                          GU  CCCG GGU UCA AA  UCC CG G CCCCGCU A
DY4480    GGGA GA GU G GCCGAGC  GGUCA AAAGCG A CAGAC UGUAAAUCUG UU  GAA     GGU   U       UUCUA CGUA GGU UCG AA  UCC UG C CUCUCCC A
DN1541    UCCG CA GU A GCUCAGU  GGU   AGAGCU A UCGGC UGUUAACCGA UCG                          GU  CGUA GGU UCG AA  UCC UA C CUGCGGA G
DR2602    GCGU CC AU C GUCUAAA  GGAU  AGGACA G AGGUU UUCUAAACCU CCA                          G   UAUA GGU UCG AA  UCC UA U UGGACGU A
DX1660    CGCG GG GU G GAGCAGCC UGGU  AGCUCG U CGGGC UCAUAACCCG AAG                          AU  CGUC GGU UCA AA  UCC GG C CCCCGCA ACCA
DS2602    GGAA AG AU G GUUGAGU  GGUUU AAGGCG U AGCAU UGGAAAUGCU AU  GUA GGCUUUU     GGUC UAU    CGAG GGU UCG AA  UCC CU C UCUUUCC G
DV2601    AGGG CU AU A GCUCAGC  GGU   AGAGCG C CUCGU UUACACCGAG AAU                          GU  CUAC GGU UCA AA  UCC GU A UAGCCCU A
DQ2920    UGGG GC GU G GCCAAGU  GGU   AAGGCA A CGGGU UUUGGUCCCG CUA                          UU  CGGA GGU UCG AA  UCC UU C CGUCCCA G
DS4362    GGAG GU AU G GCUGAGU  GGCUU AAGGCA U UGGUU UGCUAAAUCG AC  AUA CAAUAAG     AUUG UAU    CAUG GGU UCG AA  UCC CA U UUCCUCC G
DS1520    GGAG AG UU G GCAGAGC  GGU   AAUGCA G CGGAC UCGAAAUCCG CCG AGC CAAUGUU   GAAUUG GGU    CGCA GGU UCA AA  UCC UG U ACUCUCC U
DT3360    GCCC UU UU A ACUCAGU  GGU   AGAGUA A CGCCA UGGUAAGGCG UAA                          GU  CAUC GGU UCA AA  UCC GA U AAGGGGC U
DG4080    AUAG AU AU A AGUUAAGU GGU   AAACUG A AUGUC UUCCACACAU UGA                          U   UGUG AGU UCG AU  UCU CA C UAUCUAG A
DG4700    GCAU AU AU A GUAAACA  AU    AUUAUA U UUGCC UUCCAAGCAA AAG                          U   CCUA AU  AAA      AU UA G UGUAUGC U
DS6744    GUGG AC GU G CCGGAGU  GGUU  AUCGGG C AUAAC UAGAAAUCAU GU  GGG C   UUU       G CCCG   CGCA GGU UCG AA  UCC UG C CGUUCAC G
DN1350    UCCU CG GU A GCUCAAUU GGC   AGAGCA G CCGGC UGUUAACCGG CAG                          GU  UACU GGU UCG AG  UCC AG U CCGGGGA G
DQ6050    GGUC CU AU A GUGUAGU  GGUU  AUCACU U CGGAC UUUGAAUCCG AAA                          A   CCCA GGU UCG AA  UCC UG G UAGGACC A
DF5400    GUCU AC GU A GCUUAACC CCC   AAAGCA A GACAC UGAAAAUGCC UAG                          A   UGGA U U CAC AC    A UC C CAUAGAC A
DD2600    GGGA UU GU A GUUCAAUU GGUU  AGAGUA C CGCCC UGUCAAGACG GAA                          GU  UGCG GGU UCG AG  CCC CG U CAAUCCC G
DS1664    GGUG AG GU G UCCGAGU  GGCUG AAGGAG C ACGCC UGGAAAGUGU GU  AUA CG  GCA   A   CG UAU    CGGG GGU UCG AA  UCC CC C CCUCACC GCCA
DS3880    AGAG AG UU G GCUGAGU  GGU   AAGGCG A CUAGC UUGAGUCUAG UU  AAG UU  AAA       AC UUU    CAUA UGU UCG AA  UCA UA U ACUCUCU G
DP5120    CGGG AG AG A AUUUAAA  UU    AGAAUG U UGGCU UUGGGGGUCA AUA                          G   UGGA GGU UUG AG  UCC UU C UUUCUCG A
DR4800    AAAU AU GA A GCGAUUU        AUUGCA A UUAGU UUCGACCUAA UCU                          U   AGGU G   AAA UU    C AC CCCAUAUUU U
DP5200    CAAG GA AU A GUUUAU   GU    AGAAUU C CAGCU UUGGGUGUUG GUG                          G   UGAG GU  UUA AU   GU CU C UUCCUUG A
DN5160    UAGA AU GA A GCUCGCU  GGAU  AGAGUG U UUAGC UGUUAACUAA AAU                          AU  UACG GGA UCG AG  GCC CG U CAUUCUA G
DE2500    GCCC CC AU C GUCUAGUG GUUC  AGGACA U CUCUC UUUCAAGGAG GCA                          G   CGGG GAU UCG AC  UUC CC C UGGGGGU A
```

```
DZ7560    GCCC GG AUGAACCAUGGC   GGUC  UGUGGU G CAGAC UUCAAAUCUG UA  GGC G   GUU   AG   C GCCG   CAGU GGU UCG AC  UCC ACCU UUCGGGU G
DG5120    ACUU UC UU A GUAUUAA   CC    AGUACA C GUGAC UUCCAAUCAC AAA                        G   UCUU AGU AGA AU   CU AA G AGAAAGU A
DX3720    GCAG CA AG G GUGGU           CUCAAC C UGGGU UCAUUCCCCA GCU                        A   AUAA AGU UCG AU  UCU UU G UAGCGGC U
DE4980    AUUC CU GU A GUUGAA          ACAACA A UAACU UUUCAUGUUA UAG                        G   UUUA GGU UGA AC  CCC UA A CAGGAAU C
DW2720    GCGC UC UU A GUUCAGUU UGGU   AGAACG C GGGUC UCCAAAACCC GAU                        GU  CGUA GGU UCA AA  UCC UA C AGAGCGU G
DF6280    GCGG AU UU A GCUCAGUU  GGG   AGAGCG C CAGAC UGAAGAUCUG GAG                        GU  CCUG UGU UCG AU  CCA CA G AAUUCGC A
DP8041    GGCU CG UU G GUCUAGG   GGU   AUGAUU C UCGCU UCGGGUGCGA GAG                        GU  CCCG GGU UCA AA  UCC CG G ACGAGCC C
```

APPENDIX B: Part of the training data

DK0660
GGGCCCGUAGCUUAGUCUGGUAGAGCGCCUGACUUUUAAUCAGGCGGUCGAGGGUUCGAAUCCCUUCGGGCCCG
DE2520
GCCCCCAUCGUCUAGAGGCCUAGGACAUCUCCCUUUCACGGAGGCAACGGGGAUUCGAAUUCCCCUGGGGGUA
DT5320
GUCUUGAUAGUAUAAACAUUACUCUGGUCUUGUAAACCUGAAAUGAAGAUCUUCUCUUCUCAAGACA
DR1181
GCCCAUGUAGCUCAGUAGGAUAGAGCACGCGCCUUCUAAGCGUGAGGUCGGAAGUUCGAGCCUUCUCGUGGGCACCA
DC2720
GGCGGCAUGGCCAAGUGGUAAGGCAGGGGACUGCAAAUCCUUUACCCCCAGUUCAAAUCUGGGUGCCGCCU
DL5720
ACUUUUAAAGGAUAACAGCCAUCCGUUGGUCUUAGGCCCCAAAAAUUUUGGUGCAACUCCAAAUAAAAGUA
DC2680
GGCGGCAUGGCCAAGCGGUAAGGCAGGGGACUGCAAAUCCUUUAUCCCCAGUUCAAAUCUGGGUGCCGCCU
DM3920
ACCUGGGUAGUUUAAAGGUAAAACCUUAAUUUCAUACAUUAAAGAUGAGAAUUCGAUUUUCUCCCCAGGUN
DD4700
UUAAAAGUAGUUAAUAUAUAACUUAGAAUUGUCAGUUCUAGAUUCCUUUUACUAAGGCUUUUUAU
DY6280
CUCUCGGUAGCCAAGUUGGUUUAAGGCGCAAGACUGUAAAUCUUGAGAUCGGGCGUUCGACUCGCCCCCGGGAGA
DT5080
GCCUUGAAAGCUCAACAACUAGAGCUUUGGUCUUGUAAACCAGGAGAGAGGGUAAACUCCCUCUCAAGGCU
DR4000
GCUCUCUUAGCUUAAUGGUUAAAGCAUAAUACUUCUAAUAUUAAUAUUCCAUGUUCAAAUCAUGGAGAGAGUA
DL2601
GGGGGUAUGGCGAAAUUGGUAGACGCUGCGGACUUAAAAUCCGUUGGCUUUAAAGACCGUGAGGGUUCAAGUCCCUCUACCCCCA
DV6161
GGUCGGAUGGUGUAGUCGGUUAUCACGGUUGCUUUACACGCAACAGGUCUCGAGUUCGAUCCUCGGUCGGAUCA
DH4700
AAAAAAGUAGUUUAAUUUAAGAACGAUAGGUUGUGGGGCUAUUAGCGGUGUUAACCCUUUUUUA
DK4030
GAGAGUAUUGUUUAAUGGUAAAACAUUCUGUCUUUUAAGCAGUCUAUAGUGGUUCGAUUCCACUUAUUCUCA
DG2601
GCGGGUAUAGUUUAGUGGUAAAACCUUAGCCUUCCAAGCUAACGAUGCGGGUUCGAUUCCCGCUACCCGCU
DE1140
GGCCUGUUGGUGAAGCGGUUAACACACACGGUUUUCAUCCGUGGACACACGGGUUCGAACCCCGUACAGGCUACCA
DS3881
GGGAAAGUUUCCAUGGGUAGGGUAAGAUAUUUGCUAAAUAUUGUGCGUUUGCACAUAGAUGUUCGAAUCAUCUCUUUCCCG
DY5000
GAUUGGGUGGCAGAUAGGGAUGCACUAGAUUGUAAAUCUAGGUAGGAAGGUUCAAGUCCUUUCCUGGUCA
DI2580
GGGUUAUUAGCUCAGUUGGUUAGAGCACACCCCUGAUAAGGGUGAGGUCCCUGGUUCAAAUCCAGGAUAACCCA
DX4880
AAAAAGAUAAGCUAAUUAAGCUACUGGGUUCAUACCCCAUUUAUAAAGGUUAUAAUCCUUUUCUUUUUA
DD6280

UCCGUGAUAGUUUAAUGGUCAGAAUGGGCGCUUGUCGCGUGCCAGAUCGGGGUUCAAUUCCCCGUCGCGGAG
DS2920
GGAGAGAUGGCUGAGUGGACUAAAGCGGCGGAUUGCUAAUCCGUUGUACGAGUUAAUCGUACCGAGGGUUCGAAUCCCUCUCUUUCCG
DK7680
GCCCGGCUAGCUCAGUCGGUAGAGCAUGAGACUCUUAAUCUCAGGGUCGUGGGUUCGAGCCCCACGUUGGGCG
DI5120
GGAAAUGUGCCCGAAAGUAGGGAUCACUUUGAUAGAGUGAAAUAUAUGGGUUCAAACCCCAUCAUCUCCU
DI4080
GAAACUAUAAUUCAAUUGGUUAGAAUAGUAUUUUGAUAAGGUACCAAUAUAGGUUCGAUUCCUGUUAGUUUCA
DV6280
GGUUUCGUGGUCUAGUCGGUUAUGGCAUCUGCUUAACACGCAGAACGUCCCCAGUUCGAUCCUGGGCGAAAUCA
DA5040
GUGGUUCUAGUUUAGUGAAAAACGUUUGCUUUGCAAGCAGAAAUCCUAAGUUAAAUUCUUAGGAACUACA
DY6742
CCGACCUUAGCUCUGUUGGUAUAGCGGAGGACUGUAGAUCCUUAGGUCACUGGUUCGAAUUCGGUAGGUCGGA
DH2600
GGCGGACGUAGCCAAGUGGAUUAAGGCAGUGGAUUGUGGAUCCUCUACGCGCGGGUUCAAUUCCCGUCGUUCGCC
DH5880
GUAAAUAUAGUUUAACCAAAACAUCAGAUUGUGAAUCUGACAACAGAGGCUUACGACCCCUUAUUUACC
DT5220
ACUCUAAUAGUUUAUGAAAAACAUUGGUCUUGUAAACCAAAAACUGAAGACUCCACCCUUCUUAGAGUA
DI2440
GGGCUAUUAGCUCAGUUGGUUAGAGCGUUGCUUUGAUAAGGCAAAAGUCGAAAGUUCAAAUCUUUCAUAGCCCA
DS6241
GGCGCGAUGGCAGAGUGGUCUAAUGCGUGAGACUUGAAAUCUCAUUUCUUCGGAAGCGUCGGUUCAAAUCCGGCUCGCGUCG
DM6160
AGCUUCUUAACUCAGGGGUAGAGUGCGAGGCCCAUAACCUCGAGGUCCUAGGAUCGAAACCUAGAGAAGCUA
DT5360
GUCUUUGUAGUACAUCUAAUAUACUGGUCUUGUAAACCAGAGAAGGAGAACAACUAACCUCCCUAAGACU
DA0380
GGGCCCAUAGCUCAGUGGUAGAGUGCCUCCUUUGCAAGGAGGAUGCCCUGGGUUCGAAUCCCAGUGGGUCCA
DW4980
AGAGAUUUAAGUUAAUAAAACUGAAAACCUUCAAAGUUUUAAAUAAGAGUGGAACUCUCUUAGUCUUUA
DR2700
GCGUCCAUUGUCUAAUGGAUAGGACAGAGGUCUUCUAAACCUUUGGUAUAGGUUCAAAUCCUAUUGGACGCU
DH5840
GUAAAUAUAGUUUAACCAAAACAUUAGAUUGUGAAUCUAAUAAUAGGGCCCCACAACCCCUUAUUUACC
DA4500
GGGGACGUAGCUCAAUUGGUAGAGCGUAUGUUUUGCAAGCAUAAAGCUGUCGGUUCAAAUCCGAUCGUCUCCA
DQ9991
GUCUCUGUGGCGCAAUCGGUUAGCGCGUUCGGCUGUUAACCGAAAGAUUGGUGGUUCGAGCCCACCCAGGGACG
DP1140
CGGGAAGUGGCUCAGUUUGGUAGAGCAUUCGGUUUGGGACCGAAGGGUCGCAGGUUCAAAUCCUGUCUUCCCGACCA
DV0860
GGGCCCGUCGUCUAGCCUGGUUAGGACGCUGCCCUGACGCGGCAGAAAUCCUGGGUUCAAGUCCCAGCGGGCCCA
DG1500
GCGGAAGUAGUUCAGUGGUAGAACAUCACCUUGCCAUGGUGGGGGUCGCGGGUUCGAAUCCCGUCUUCCGCU
DG2681

GCGGGUAUAGUUUAGUGGUAAAACCCUAGCCUUCCAAGCUAACGAUGCGGGUUCGAUUCCCGCUACCCGCU
DP1662
CGGCACGUAGCGCAGCCUGGUAGCGCACCGUCAUGGGGUGUCGGGGGUCGGAGGUUCAAAUCCUCUCGUGCCGACCA
DF5920
GUUAAUGUAGCUUUAAAAUUAAAGCAAGGCACUGAAAAUGCCUAGAUGAGUGCUCCAACUCCAUAAACA
DN5000
UGGGUUGUAGCCUAAUGGAAAGGCAAUUGGCCGUUAACCAGGAGAUAACAAGAUCAAUACUUGUCAACUCAG
DG4501
GCGGAUAUAGAUUAAAGGUAAAUUAUCUGCCUUCCAAGCAGAGGAUAUGGGUUCGAUUCCCGUUAUCCGCA
DC4840
GGUCUUAUAGUCAAUAAUGAUAUCAAACUGCAAUUUUGAAGGAGUAAGUUUUACUAAGGCUU
DC5160
GGCCUUGGGGUGUCAACACGUGGGGUUGCAAACCCCAAGAUGCAGUAUAAUACCUGCCGGGGCUU
DD1180
GGCCCCAUAGCGAACGUUGGUUAUCGCGCCUCCCUGUCACGGAGGAGAUCACGGGUUCGAGUCCCGUUGGGGUCGCCA
DA2520
GGGGGUAUAGCUCAGUUGGUAGAGCGCUGCCUUUGCAAGGCAGAUGUCAGCGGUUCGAGUCCGCUUAUCUCCA
DN6060
GCUCGAUUAGCUCAGCUGGUUAGAGCAUGCGGCUGUUAACCGCAAGGUCGUAGGUUCGAUCCCUACAUCGAGCG
DF5970
GUUCAUGUAGCUUAAAACCAAAGCAAGGCAUUGAAAAUGCCUAGAUGAGUAUAUUAACUCCAUAAACA
DD1140
GGCCCCAUAGCGAAGUUGGUUAUCGCGCCUCCCUGUCACGGAGGAGAUCACGGGUUCGAGUCCCGUUGGGGUCGCCA
DF7740
GCCGAAAUAGCUCAGUUGGGAGAGCGUUAGACUGAAGAUCUAAAGGUCCCCGGUUCAAUCCCGGGUUUCGGCA
DW5040
AAGAGCUUAAGUUAAAUAAACUGAAAGCCUUCAAAGCUUUUUAUAAGAAUGGAAACUUCUUAGCUCUUG
DF4700
GAUACGGUAGCUUAAUUAAAGCGUCUCAUUGAAAAUGAGGAAGAUGGUACUUUAGUACCUUGUGUCA
DP7560
GGCCGAAUGGUCUAGUGGUAUGAUUCUCGCUUUGGGUGCGAGAGGUCCCGGGUUCAAUCCCCGGUUCGGCCC
DW3960
AAGAGUAUAGUUUAAAGGUAAAACAGAAAGCUUCAACCUUUAAUUUCUUAGUUCGAGUCUAAGUGCUCUUG
DA2540
GGGGGUAUAGCUCAGUUGGUAGAGCGCUGCCUUUGCACGGCAGAUGUCAGGGGUUCGAGUCCCCUUACCUCCA
DA1540
GGGGCCUUAGCUCAGCUGGGAGAGCGCCUGCUUUGCACGCAGGAGGUCAGCGGUUCGAUCCCGCUAGGCUCCA
DQ4440
UGGAGUAUAGCCAAGUGGUAAGGCAUCGGUUUUUGGUAUCGGCAUGCAAAGGUUCGAAUCCUUUUACUCCAG
DL5560
ACUUUUAAAGGAUAACAGCUAUCCAUUGGCCUUAGGAGUCAAAAAUAUUGGUGCAACUCCAAAUAAAAGUA
DL2180
GGGGAUGUGGUGGAAUUGGUAGACGCAACGGACUUAAAAUCCGUCGAUUGUAUAGAUCGUGAGGGUUCAAGUCCCUCCGUCCCCA
DX0980
AGCGGGGUAGGCCAGCCAGGUAGGCCGCGGGGCUCAUANCCCCGAGGUCCCGGGUUCAAAUCCCGGCCCCGCUA
DY4480
GGGAGAGUGGCCGAGCGGUCAAAAGCGACAGACUGUAAAUCUGUUGAAGGUUUUCUACGUAGGUUCGAAUCCUGCCUCUCCCA
DN1541

UCCGCAGUAGCUCAGUGGUAGAGCUAUCGGCUGUUAACCGAUCGGUCGUAGGUUCGAAUCCUACCUGCGGAG
DR2602
GCGUCCAUCGUCUAAAGGAUAGGACAGAGGUUUUCUAAACCUCCAGUAUAGGUUCGAAUCCUAUUGGACGUA
DX1660
CGCGGGGUGGAGCAGCCUGGUAGCUCGUCGGGCUCAUAACCCGAAGAUCGUCGGUUCAAAUCCGGCCCCCGCAACCA
DS2602
GGAAAGAUGGUUGAGUGGUUUAAGGCGUAGCAUUGGAAAUGCUAUGUAGGCUUUUGGUCUAUCGAGGGUUCGAAUCCCUCUCUUUCCG
DV2601
AGGGCUAUAGCUCAGCGGUAGAGCGCCUCGUUUACACCGAGAAUGUCUACGGUUCAAAUCCGUAUAGCCCUA
DQ2920
UGGGGCGUGGCCAAGUGGUAAGGCAACGGGUUUUGGUCCCGCUAUUCGGAGGUUCGAAUCCUUCCGUCCCAG
DS4362
GGAGGUAUGGCUGAGUGGCUUAAGGCAUUGGUUUGCUAAAUCGACAUACAAUAAGAUUGUAUCAUGGGUUCGAAUCCCAUUUCCUCCG
DS1520
GGAGAGUUGGCAGAGCGGUAAUGCAGCGGACUCGAAAUCCGCCGAGCCAAUGUUGAAUUGGGUCGCAGGUUCAAAUCCUGUACUCUCCU
DT3360
GCCCUUUUAACUCAGUGGUAGAGUAACGCCAUGGUAAGGCGUAAGUCAUCGGUUCAAAUCCGAUAAGGGGCU
DG4080
AUAGAUAUAAGUUAAGUGGUAAACUGAAUGUCUUCCACACAUUGAUUGUGAGUUCGAUUCUCACUAUCUAGA
DG4700
GCAUAUAUAGUAAACAAUAUUAUAUUUGCCUUCCAAGCAAAAGUCCUAAUAAAAUUAGUGUAUGCU
DS6744
GUGGACGUGCCGGAGUGGUUAUCGGGCAUAACUAGAAAUCAUGUGGGCUUUGCCCGCGCAGGUUCGAAUCCUGCCGUUCACG
DN1350
UCCUCGGUAGCUCAAUUGGCAGAGCAGCCGGCUGUUAACCGGCAGGUUACUGGUUCGAGUCCAGUCCGGGGAG
DQ6050
GGUCCUAUAGUGUAGUGGUUAUCACUUCGGACUUUGAAUCCGAAAACCCAGGUUCGAAUCCUGGUAGGACCA
DF5400
GUCUACGUAGCUUAACCCCCAAAGCAAGACACUGAAAAUGCCUAGAUGGAUUCACACAUCCCAUAGACA
DD2600
GGGAUUGUAGUUCAAUUGGUUAGAGUACCGCCCUGUCAAGACGGAAGUUGCGGGUUCGAGCCCCGUCAAUCCCG
DS1664
GGUGAGGUGUCCGAGUGGCUGAAGGAGCACGCCUGGAAAGUGUGUAUACGGCAACGUAUCGGGGGUUCGAAUCCCCCCCUCACCGCCA
DS3880
AGAGAGUUGGCUGAGUGGUAAGGCGACUAGCUUGAGUCUAGUUAAGUUAAAACUUUCAUAUGUUCGAAUCAUAUACUCUCUG
DP5120
CGGGAGAGAAUUUAAAUUAGAAUGUUGGCUUUGGGGGUCAAUAGUGGAGGUUUGAGUCCUUCUUUCUCGA
DR4800
AAAUAUGAAGCGAUUUAUUGCAAUUAGUUUCGACCUAAUCUUAGGUGAAAUUCACCCCAUAUUUU
DP5200
CAAGGAAUAGUUUAUGUAGAAUUCCAGCUUUGGGUGUUGGUGGUGAGGUUUAAUGUCUCUUCCUUGA
DN5160
UAGAAUGAAGCUCGCUGGAUAGAGUGUUUAGCUGUUAACUAAAAUAUUACGGGAUCGAGGCCCGUCAUUCUAG
DE2500
GCCCCCAUCGUCUAGUGGUUCAGGACAUCUCUCUUUCAAGGAGGCAGCGGGGAUUCGACUUCCCCUGGGGGUA
DZ7560
GCCCGGAUGAACCAUGGCGGUCUGUGGUGCAGACUUCAAAUCUGUAGGCGGUUAGCGCCGCAGUGGUUCGACUCCACCUUUCGGGUG
DG5120

ACUUUCUUAGUAUUAACCAGUACACGUGACUUCCAAUCACAAAGUCUUAGUAGAAUCUAAGAGAAAGUA
DX3720
GCAGCAAGGGUGGUCUCAACCUGGGUUCAUUCCCCAGCUAAUAAAGUUCGAUUCUUUGUAGCGGCU
DE4980
AUUCCUGUAGUUGAAACAACAAUAACUUUUCAUGUUAUAGGUUUAGGUUGAACCCCUAACAGGAAUC
DW2720
GCGCUCUUAGUUCAGUUUGGUAGAACGCGGGUCUCCAAAACCCGAUGUCGUAGGUUCAAAUCCUACAGAGCGUG
DF6280
GCGGAUUUAGCUCAGUUGGGAGAGCGCCAGACUGAAGAUCUGGAGGUCCUGUGUUCGAUCCACAGAAUUCGCA
DP8041
GGCUCGUUGGUCUAGGGGUAUGAUUCUCGCUUCGGGUGCGAGAGGUCCCGGGUUCAAAUCCCGGACGAGCCC

APPENDIX C: The comparison between our Parallel Single-SCFG Training model and COVE

Output from our SCFG constructed in the Parallel Single-SCFG Training model

```
#=CS      >>>>>>>..>>>>.><.....<<<<.>>>>>.....<<<<<..>....<.....>>>>>..><...<<<<<<<<<<<....
DF6280    GCGGAUUUAGCUCAGUU.GGGA.GAGCGCCAGACUGAA.....GAUCUGGAGG....U.....CCUGUGUUCGAUCCACAGAAUUCGCAcca
#=SS      >>>>>>>..>>>>.><>...<..<<<<.>>>>>.....<<<<<..>....<.....>>>>>..><...<<<<<<<<<<<....
DF6280G   GCGGATTTAGCTCAGTT.GGGA.GAGCGCCAGACTGAAgaaatacttcggtcaagtt.ATCTGGAGG....T.....CCTGTGTTCGATCCACAGAATTCGCA...
#=SS      >>>>>>>..>>>>.><>...<..<<<<.>>>>>.....<<<<<..>....<.....>>>>>..><...<<<<<<<<<<<....
DD6280    UCCGUGAUAGUUUAAU..GGUCAGAAUGGGCGCUUGUC.....GCGUGCCAGA.....UCGGGGUUCAAUUCCCCGUCGCGGAGcca
#=SS      >>>>>>>..>>>>.><.....<<<<.>>>>>..><>.....<.<<<<<.....>>>>>.....<<<<<<<<<<<....
DX1661    CGCGGGGUGGAGCAGCCuGGUA.GCUCGUCGGGCUCAU.....AACCCGAAGG....U.....CGUCGGUUCAAAUCCGGCCCCCGCAAcca
#=SS      .>>>>>>..>>>>.><.....<<<<.>>>>>....>.....<.<<<<<..>....<.....>>>>>.>...<.<<<<<<<<<<<.....
DS6280    GGCAACUUGGCCGAGU..GGUUAAGGCGAAAGAUUAGA.....AAUCUUUUGGgcuuUgcccgCGCAGGUUCGAGUCCUGCAGUUGUCGcca
#=SS      >>>>>>>..>>>..><.....><.<<<>>>>>>>.....<<<<<<<.>....<.....>>>>>.>><.<.<<<<<<<<<<<....

output from COVE

#=CS      >>>>>>>..>>>>.><.....<<<<.>>>>>.....<<<<<..>....<.....>>>>>..><...<<<<<<<<<<<....
DF6280    GCGGAUUUAGCUCAGUU.GGGA.GAGCGCCAGACUGAA.....GAUCUGGAGG....U.....CCUGUGUUCGAUCCACAGAAUUCGCAcca
#=SS      >>>>>>>..>>>>.><>...<..<<<<.>>>>>.....<<<<<..>....<.....>>>>>..><...<<<<<<<<<<<....
DF6280G   GCGGATTTAGCTCAGTT.GGGA.GAGCGCCAGACTGAAgaaatacttcggtcaagtt.ATCTGGAGG....T.....CCTGTGTTCGATCCACAGAATTCGCA...
#=SS      >>>>>>>..>>>>.><>...<..<<<<.>>>>>.....<<<<<..>....<.....>>>>>..><...<<<<<<<<<<<....
DD6280    UCCGUGAUAGUUUAAU..GGUCAGAAUGGGCGCUUGUC.....GCGUGCCAGA.....UCGGGGUUCAAUUCCCCGUCGCGGAGcca
#=SS      >>>>>>>..>>>>.><.....<<<<.>>>>>..><>.....<.<<<<<.....>>>>>.....<<<<<<<<<<<....
DX1661    CGCGGGGUGGAGCAGCCuGGUA.GCUCGUCGGGCUCAU.....AACCCGAAGG....U.....CGUCGGUUCAAAUCCGGCCCCCGCAAcca
#=SS      .>>>>>>..>>>>.><.....<<<<.>>>>>....>.....<.<<<<<..>....<.....>>>>>.>...<.<<<<<<<<<<<.....
DS6280    GGCAACUUGGCCGAGU..GGUUAAGGCGAAAGAUUAGA.....AAUCUUUUGGgcuuUgcccgCGCAGGUUCGAGUCCUGCAGUUGUCGcca
#=SS      >>>>>>>..>>>..><.....><.<<<>>>>>>>.....<<<<<<<.>....<.....>>>>>.>><.<.<<<<<<<<<<<....
```

From the above test results, the SCFG constructed from the parallel Single-SCFG training model produce exactly same result comparison with COVE.

APPENDIX D: Multiple alignments and its related consensus structure sequences after the Sub-SCFGs are stable

For simplest, this appendix just present 10 multiple alignment, each of them contains ten sequences, from 10 Sub-SCFG.

```
Sub-SCFG0 aligned sequences:
seqs[0]: .GGGCCCGUAGCUUAGUcuGGUA..GAGCGCCUGACUUUUAAUCAGGCGGUCGAGGGUUCGAAUCCCUUCGGGCCCG
seqs[1]: .GCCUUGAAAGCUCAAC..AACU.AGAGCUUUGGUCUUGUAAACCAGGAG.AGAGGGU.AAACUCCCUCUCAAGGCU
seqs[2]: .GGGUUAUUAGCUCAGUu.GGUU.AGAGCACACCCCUGAUAAGGGUGAGGUCCCUGGUUCAAAUCCAGGAUAACCCA
seqs[3]: gGCGGACGUAGCCAAGU..GGAUuAAGGCAGUGGAUUGUGGAUCCUCUACGCGCGGGUUCAAUUCCCGUCGUUCGCC
seqs[4]: .GUAAAUAUAGUUUAAC....CA..AAACAUUAGAUUGUGAAUCUAAUAA.UAGGGCCCCACAACCCCUUAUUUACC
seqs[5]: .GCGGAUAUAGAUUAAA..GGUA..AAUUAUCUGCCUUCCAAGCAGAGGA.UAUGGGUUCGAUUCCCGUUAUCCGCA
seqs[6]: .GAUACGGUAGCUUAAU....UA..AAGCGUCUCAUUGAAAAUGAGGAAGAUGGUA.CUUUAG..UACCUUGUGUCA
seqs[7]: .UCCGCAGUAGCUCAGU..GGUA..GAGCUAUCGGCUGUUAACCGAUCGGUCGUAGGUUCGAAUCCUACCUGCGGAG
seqs[8]: .GCAUAUAUAGUAAACA..AUA...UUAUAUUUGCCUUCCAAGCAAAAGU.CCUAA..UAAAA..UUAGUGUAUGCU
seqs[9]: .CAAGGAAUAGUUUAUG....UA..GAAUUCCAGCUUUGGGUGUUGGUGG.UGAGGUUU..AAUGUCUCUUCCUUGA

Consensus secondary structure:
         .>>>>>>>...>>>.><....><..<<<>.>>>>>.......<<<<<.<...>>>>>.>..<..<<<<<<<<<<<<.

Sub-SCFG1 aligned sequences:
seqs[0]: GCCCCCAUCGUCUAGA.GGCCu...AGGACAUCUCCCUUUCACGGAGGCAA..........CGGGGAUUCGAAUUCCCCUGGGGGUA
seqs[1]: GCUCUCUUAGCUUAAU.GGUU....AAAGCAUAAUACUUCUAAUAUUAAUA..........UUCCAUGUUCAAAUCAUGGAGAGAGUA
seqs[2]: AAAAAGAUAAGCUAAU...UA.....AGCUACUGGGUUCAUACCCCAUUUA..........UAAAGGUUAUAAUCCUUUUCUUUUUA
seqs[3]: GUAAAUAUAGUUUAA....CC....AAAACAUCAGAUUGUGAAUCUGACAA..........CAGAGGCUUACGACCCCUUAUUUACC
seqs[4]: GGGGACGUAGCUCAAUuGGUA.....GAGCGUAUGUUUUGCAAGCAUAAAG..........CUGUCGGUUCAAAUCCGAUCGUCUCCA
seqs[5]: GGUCUUAUAGUCAAU....A.....AUGAUAUCAAACUGCAAUUUUGAAGG..........AGUAA..GUU....UUACUAAGGCUU
seqs[6]: GGCCGAAUGGUCUAGU.GGUA.....UGAUUCUCGCUUUGGGUGCGAGAGG..........UCCCGGGUUCAAUCCCCGGUUCGGCCC
seqs[7]: GCGUCCAUCGUCUAAA.GGAU....AGGACAGAGGUUUUCUAAACCUCCAG..........UAUAGGUUCGAAUCCUAUUGGACGUA
seqs[8]: GUGGACGUGCCGGAGU.GGUU....AUCGGGCAUAACUAGAAAUCAUGUGGgcuuugcccg.CGCAGGUUCGAAUCCUGCCGUUCACG
seqs[9]: UAGAAUGAAGCUCGCU.GGAU....AGAGUGUUUAGCUGUUAACUAAAAUA..........UUACGGGAUCGAGGCCCGUCAUUCUAG

Consensus secondary structure:
         .>>>>>>.>.>>>............<<<..>>>>>.......<<<<<.............<>>>>>>.><><<<<<<<<<<<<..

Sub-SCFG2 aligned sequences:
seqs[0]: GUCUUGAUAGUAUAA...AC.A..UUACUCUGGUCUUGUAAACCUGAAA............UGAAGAUCUU..CUCUUCUCAAGACA...
seqs[1]: GGGGGUAUGGCGAAAUU.GGUA.GACGCUGCGGACUUAAAAUCCGUUGGcuuuaaagaccg.UGAGGGUUCAAGUCCCUCUACCCCCA...
```

```
seqs[2]: UCCGUGAUAGUUUAAU..GGUC.AGAAUGGGCGCUUGUCGCGUGCCAGA.............UCGGGGUUCAAUUCCCCGUCGCGGAG...
seqs[3]: ACUCUAAUAGUUUAU...GA.A..AAACAUUGGUCUUGUAAACCAAAAA.............CUGAAGACUCCACC.CUUCUUAGAGUA...
seqs[4]: GUCUCUGUGGCGCAAUC.GGUU.AGCGCGUUCGGCUGUUAACCGAAAGA.............UUGGUGGUUCGAGCCCACCCAGGGACG...
seqs[5]: GGCCUUGGGGUGUC.......A..ACACGUGGGGUUGCAAACCCCAAGA.............UGCAGUAUAAUAC.CUGCCGGGGCUU...
seqs[6]: AAGAGUAUAGUUUAAA..GGUA..AAACAGAAAGCUUCAACCUUUAAUU.............UCUUAGUUCGAGUCUAAGUGCUCUUG...
seqs[7]: CGCGGGGUGGAGCAGCCuGGUA..GCUCGUCGGGCUCAUAACCCGAAGA.............UCGUCGGUUCAAAUCCGGCCCCCGCAAcca
seqs[8]: UCCUCGGUAGCUCAAUU.GGCA..GAGCAGCCGGCUGUUAACCGGCAGG.............UUACUGGUUCGAGUCCAGUCCGGGGAG...
seqs[9]: GCCCCCAUCGUCUAGU..GGUUcAGGACAUCUCUCUUUCAAGGAGGCAG.............CGGGGAUUCGACUUCCCCUGGGGGUA...
Consensus secondary structure:
          >>>>>>>.>.>>>.><........<<<..>>>>>.......<<<<..<.............<>>>>>.......<<<<<<<<<<<<....


Sub-SCFG3 aligned sequences:
seqs[0]: GCCCAUGU..AGCUCA.GUAGGA.U.AGAGCACGCGCCUUCUAAGCGUGA.................GGUCGGAAGUUCGAGCCUUC.UCGUGGGCACCA
seqs[1]: GGUCGGAU..GGUGUA.GUCGGU.U.AUCACGGUUGCUUUACACGCAACA.................GGUCUCGAGUUCGAUCCUCG.GUCGGAUCA...
seqs[2]: GGAGAGAU..GGCUGA.GU.GGA.CuAAAGCGGCGGAUUGCUAAUCCGUUGUACGAGUU.A.Aucg.UACCGAGGGUUCGAAUCCCU.CUCUUUCCG...
seqs[3]: GGGCUAUU..AGCUCA.GUUGGU.U.AGAGCGUUGCUUUGAUAAGGCAAA.................AGUCGAAAGUUCAAAUCUUU.CAUAGCCCA...
seqs[4]: CGGGAAGU..GGCUCA.GUUUGG.U.AGAGCAUUCGGUUUGGGACCGAAG.................GGUCGCAGGUUCAAAUCCUG.UCUUCCCGACCA
seqs[5]: GGCCCCAU..AGCGAAcGUUGGU.U.AUCGCGCCUCCCUGUCACGGAGGA.................GAUCACGGGUUCGAGUCCCG.UUGGGGUCGCCA
seqs[6]: GGGGGUAU..AGCUCA.GU.UGG.U.AGAGCGCUGCCUUUGCACGGCAGA.................UGUCAGGGGUUCGAGUCCCC.UUACCUCCA...
seqs[7]: GGAAAGAU..GGUUGA.GU.GGU.UuAAGGCGUAGCAUUGGAAAUGCUAUGUAGGCUUU.U.Gguc.UAUCGAGGGUUCGAAUCCCU.CUCUUUCCG...
seqs[8]: GGUCCUAU..AGUGUA.GU.GGU.U.AUCACUUCGGACUUUGAAUCCGAA.................AA.CCCAGGUUCGAAUCCUG.GUAGGACCA...
seqs[9]: GCCCGGAUgaACCAUG.GC.GGU.C.UGUGGUGCAGACUUCAAAUCUGUAGGCGGUUAG.C.G....CCGCAGUGGUUCGACUCCACcUUUCGGGUG...

Consensus secondary structure:
         >>>>>>>....>>>>.....>.<....<<<<.>>>>>.......<<<<<.................><.>>>>>.....><<<<<.<<<<<<<<<....

Sub-SCFG4 aligned sequences:
seqs[0]: GGCGGCAUGGCCAAG.U.GG.UA..AGGCAGGGGACUGCAAAUCCUUU.........AC.CCCCAGUUCAAAUCUGGG.UGCCGCCU
seqs[1]: AAAAAAGUAGUUUAA.U.UU.AA..GAACGAUAGGUUGUGGGGCUAUU.........AG.CGGU...GUUA....ACC.CUUUUUUA
seqs[2]: GCCCGGCUAGCUCAG.UCGG.UA..GAGCAUGAGACUCUUAAUCUCAG.........GGUCGUGGGUUCGAGCCCCAC.GUUGGGCG
seqs[3]: GGCGCGAUGGCAGAG.U.GG.UCuaAUGCGUGAGACUUGAAAUCUCAUuucuucggaAG.CGUCGGUUCAAAUCCGGC.UCGCGUCG
seqs[4]: GGGCCCGUCGUCUAGcCUGGuUA..GGACGCUGCCCUGACGCGGCAGA.........AAUCCUGGGUUCAAGUCCCAG.CGGGCCCA
seqs[5]: GGGGGUAUAGCUCAG.UUGG.UA..GAGCGCUGCCUUUGCAAGGCAGA.........UGUCAGCGGUUCGAGUCCGCU.UAUCUCCA
seqs[6]: GGGGCCUUAGCUCAG.CUGG.GA..GAGCGCCUGCUUUGCACGCAGGA.........GGUCAGCGGUUCGAUCCCGCU.AGGCUCCA
seqs[7]: AGGGCUAUAGCUCAG.C.GG.UA..GAGCGCCUCGUUUACACCGAGAA.........UGUCUACGGUUCAAAUCCGUA.UAGCCCUA
seqs[8]: GUCUACGUAGCUUAA.CCCC.CA..AAGCAAGACACUGAAAAUGCCUA.........GA.UGGAU.UCACAC..AUCC.CAUAGACA
seqs[9]: ACUUUCUUAGUAUUA.A.CC..A..GUACACGUGACUUCCAAUCACAA.........AG.UCUUAGUAGAA.UCUAAG.AGAAAGUA

Consensus secondary structure:
          >>>>>>>...>>..>......<....<<..>>>>>..><...<<<<<.............>>>>>.......<<<<<.<<<<<<<.

Sub-SCFG5 aligned sequences:
```

```
seqs[0]: ACUUUUAAAGGAUAAC.AGCC..AUCCG.UUGGUCUUAGGCCCCAAAAA.UUUUGGUGCAACUCCAAAUAAAAGUA
seqs[1]: GAGAGUAUUGUUUAAU.GGUA..AAACAuUCUGUCUUUUAAGCAGUCUA.UAGUGGUUCGAUUCCACUUAUUCUCA
seqs[2]: GGAAAUGUGCCCGAAA....GU.AGGGA.UCACUUUGAUAGAGUGAAAUAUAUGGGUUCAAACCCCAUCAUCUCCU
seqs[3]: AGCUUCUUAACUCAGG.GGUA..GAGUG.CGAGGCCCAUAACCUCGAGGUCCUAGGAUCGAAACCUAGAGAAGCUA
seqs[4]: GCGGAAGUAGUUCAGU.GGUA..GAACA.UCACCUUGCCAUGGUGGGGGUCGCGGGUUCGAAUCCCGUCUUCCGCU
seqs[5]: GCUCGAUUAGCUCAGCuGGUUA.GAGCA.UGCGGCUGUUAACCGCAAGGUCGUAGGUUCGAUCCCUACAUCGAGCG
seqs[6]: UGGAGUAUAGCCAAGU.GGUA..AGGCA.UCGGUUUUUGGUAUCGGCAUGCAAAGGUUCGAAUCCUUUUACUCCAG
seqs[7]: UGGGGCGUGGCCAAGU.GGUA..AGGCA.ACGGGUUUUGGUCCCGCUAUUCGGAGGUUCGAAUCCUUCCGUCCCAG
seqs[8]: GGGAUUGUAGUUCAAUuGGUUA.GAGUA.CCGCCCUGUCAAGACGGAAGUUGCGGGUUCGAGCCCCGUCAAUCCCG
seqs[9]: GCAGCAAGGGUG..GU....CU..CAAC.CUGGGUUCAUUCCCCAGCUA.AUAAAGUUCGAUUCUUUGUAGCGGCU

Consensus secondary structure:
        >>>>>>>..>>>>....>.<...<<<<..>>>>>.......<<<<<.....>>>>>.>...<.<<<<<<<<<<<<.

Sub-SCFG6 aligned sequences:
seqs[0]: GGCGGCAUGGCCAAGC.GGUA..AGGCAGGGGACUGCAAAUCCUUU..............AU.CCCCAGUUCAAAU...CUGGGUGCCGCCU...
seqs[1]: GCGGGUAUAGUUUAGU.GGUA..AAACCUUAGCCUUCCAAGCUAAC..............GA.UGCGGGUUCGAUU...CCCGCUACCCGCU...
seqs[2]: GAAACUAUAAUUCAAUuGGUU.AGAAUAGUAUUUUGAUAAGGUACC..............AA.UAUAGGUUCGAUU...CCUGUUAGUUUCA...
seqs[3]: GUCUUUGUAGUACAU...CUA..AUAUACUGGUCUUGUAAACCAGA..............GA.AGGAGAA.CAACU.aaCCUCCCUAAGACU...
seqs[4]: GCGGGUAUAGUUUAGU.GGUA..AAACCCUAGCCUUCCAAGCUAAC..............GA.UGCGGGUUCGAUU...CCCGCUACCCGCU...
seqs[5]: GUUCAUGUAGCUUAAA.ACCA..AAGCAAGGCAUUGAAAAUGCCUA..............GA.UGAGUA.U..AUU...AACUCCAUAAACA...
seqs[6]: ACUUUUAAAGGAUAAC.AGCU..AUCCAUUGGCCUUAGGAGUCAAA..............AA.UAUUGGUGCAACU...CCAAAUAAAAGUA...
seqs[7]: GGAGGUAUGGCUGAGU.GGCUuAAGGCAUUGGUUUGCUAAAUCGACauacaauaagauuguAU.CAUGGGUUCGAAU...CCCAUUUCCUCCG...
seqs[8]: GGUGAGGUGUCCGAGU.GGCUgAAGGAGCACGCCUGGAAAGUGUGUauacggcaacg....UAuCGGGGGUUCGAAU...CCCCCCCUCACCGcca
seqs[9]: AUUCCUGUAGUUGAA....A.C..AACAAUAACUUUUCAUGUUAUA..............GG.UUUAGGUUGAACC...CCUAACAGGAAUC...

Consensus secondary structure:
        >>>>>>>.>>>>...>.<......<<<.>>>>>.......<<<<<.................<>>>>><.........<<<<<<<<<<....

Sub-SCFG7 aligned sequences:
seqs[0]: ACCUGGGUAGUUUAAA..GGUA..AAACCUUAAUUUCAUACAUUAAAGA.................U.GAGAAUUCGAUUUUCUC..CCCAGGUA...
seqs[1]: GGCCUGUUGGUGAAGC..GGUUA.ACACACACGGUUUUCAUCCGUGGAC.................ACACGGGUUCGAACCCCGU..ACAGGCUACCA
seqs[2]: GGUUUCGUGGUCUAGUC.GGUUA.UGGCAUCUGCUUAACACGCAGAACG.................UCCCCAGUUCGAUCCUGGG..CGAAAUCA...
seqs[3]: GGGCCCAUAGCUCAGU..GGUA..GAGUGCCUCCUUUGCAAGGAGGAUG.................CCCUGGGUUCGAAUCCCAG..UGGGUCCA...
seqs[4]: CGGCACGUAGCGCAGCCuGGUA..GCGCACCGUCAUGGGGUGUCGGGGG.................UCGGAGGUUCAAAUCCUCU..CGUGCCGACCA
seqs[5]: GGCCCCAUAGCGAAGUU.GGUUA.UCGCGCCUCCCUGUCACGGAGGAGA.................UCACGGGUUCGAGUCCCGU..UGGGGUCGCCA
seqs[6]: GGGGAUGUGGUGGAAUU.GGUAG.ACGCAACGGACUUAAAAUCCGUCGAUUGUAUAGAUc......GUGAGGGUUCAAGUCCCUC..CGUCCCCA...
seqs[7]: GGAGAGUUGGCAGAGC..GGUA..AUGCAGCGGACUCGAAAUCCGCCGAGCCAAUGUUGaauugggUCGCAGGUUCAAAUCCUGU..ACUCUCCU...
seqs[8]: AGAGAGUUGGCUGAGU..GGUA..AGGCGACUAGCUUGAGUCUAGUUAAGUUAAAACUU.......UCAUAUGUUCGAAUCAUAU..ACUCUCUG...
seqs[9]: GCGCUCUUAGUUCAGUUuGGUA..GAACGCGGGUCUCCAAAACCCGAUG.................UCGUAGGUUCAAAUCCUAC..AGAGCGUG...

Consensus secondary structure:
        >>>>>>>.>.>>..><.........<<<.>>>>>.......<<<<<.....................>>>>>.......<<<<<..<<<<<<<....
```

```
Sub-SCFG8 aligned sequences:
seqs[0]: UUAAAAGUAGUUAAUA.U...A.UAACUUAGAAUUGUCAGUUCUAG.........AU.UCCUU..UU.ACU.AAGGCUUUUUAU
seqs[1]: GGGAAAGU.UUCCAUG..GGUAgGGUAAGAUAUUUGCUAAAUAUUGugcguuugcaCA.UAGAUGUUCGAAUCAUCUCUUUCCCG
seqs[2]: GUGGUUCUAGUUUAGU..GA.A.AAACGUUUGCUUUGCAAGCAGAA.........AU.CCUAAGUUAAAUUCUUAGGAACUACA
seqs[3]: AGAGAUUUAAGUUAA...UA.A.AACUGAAAACCUUCAAAGUUUUA.........AA.UAAGAGUGGAACUCUCUUAGUCUUUA
seqs[4]: GUUAAUGUAGCUUUAA..AAUUaAAGCAAGGCACUGAAAAUGCCUA.........GA.UGAGUGCUCCAA...CUCCAUAAACA
seqs[5]: GCCGAAAUAGCUCAGUU.GGGA.GAGCGUUAGACUGAAGAUCUAAA.........GGUCCCCGGUUCAAUCCCGGGUUUCGGCA
seqs[6]: AGCGGGGUAGGCCAGCCaGGUA.GGCCGCGGGGCUCAUAACCCCGA.........GGUCCCGGGUUCAAAUCCCGGCCCCGCUA
seqs[7]: GCCCUUUUAACUCAGU..GGUA.GAGUAACGCCAUGGUAAGGCGUA.........AGUCAUCGGUUCAAAUCCGAUAAGGGGCU
seqs[8]: CGGGAGAGAAUUUAA...AUUA.GAAUGUUGGCUUUGGGGGUCAAU.........AG.UGGAGGUUUGAGUCCUUCUUUCUCGA
seqs[9]: GCGGAUUUAGCUCAGUU.GGGA.GAGCGCCAGACUGAAGAUCUGGA.........GGUCCUGUGUUCGAUCCACAGAAUUCGCA

Consensus secondary structure:
         >>>>>>>..>>>>.><.......<<<<.>>>>>.......<<<<<.............>>>>>>..<...<<<<<<<<<<<.


Sub-SCFG9 aligned sequences:
seqs[0]: CUCUCGGUAGCCAAGUUGGUUuAAGGCGCAAGACUGUAAAUCUUGAGA...........UCGGGCGUUCGACUCGCCCCCGGGAGA
seqs[1]: GAUUGGGUGGCAGAUA.GG.G..AUGCACUAGAUUGUAAAUCUAGGUA...........GGAAGGUUCAAGUCCUUUCCUGGUCA
seqs[2]: CCGACCUUAGCUCUGUUGGUA..UAGCGGAGGACUGUAGAUCCUUAGG...........UCACUGGUUCGAAUUCGGUAGGUCGGA
seqs[3]: GCGUCCAUUGUCUAAU.GGAU.AGGACAGAGGUCUUCUAAACCUUUGG...........UAUAGGUUCAAAUCCUAUUGGACGCU
seqs[4]: UGGGUUGUAGCCUAAU.GGAA..AGGCAAUUGGCCGUUAACCAGGAGA...........UAACAAGAUCAAUACUUGUCAACUCAG
seqs[5]: AAGAGCUUAAGUUAAA....U.AAACUGAAAGCCUUCAAAGCUUUUUA...........UAAGAAUGGAAACUUCUUAGCUCUUG
seqs[6]: GGGAGAGUGGCCGAGC.GGUCaAAAGCGACAGACUGUAAAUCUGUUGAagguuuucua..CGUAGGUUCGAAUCCUGCCUCUCCCA
seqs[7]: AUAGAUAUAAGUUAAGUGGUA..AACUGAAUGUCUUCCACACAUUGAU...........UGUGAGUUCGAUUCUCACUAUCUAGA
seqs[8]: AAAUAUGAAGCGAUUU....A..UUGCAAUUAGUUUCGACCUAAUCU...........UAGGUGAA..AUUCACCCCAUAUUUU
seqs[9]: GGCUCGUUGGUCUAGG.GGUA..UGAUUCUCGCUUCGGGUGCGAGAGG...........UCCCGGGUUCAAAUCCCGGACGAGCCC

Consensus secondary structure:
         >>>>>>>..><.><>.........><.>>>>>..><...<<<<<.............<>>>>>>....<<.<<<<<<<<<<.
```

APPENDIX E: Initial consensus secondary structure sequences and aligned sequences

Unaligned consensus secondary structure sequences from ten Sub-SCFGs

```
Consensus 0: .>>>>>>>...>>>.><....><..<<<>.>>>>>.......<<<<<.<...>>>>>.>..<..<<<<<<<<<<<<.

Consensus 1:.>>>>>>.>.>>>.............<<<..>>>>>.......<<<<<.............<>>>>>>.><><<<<<<<<<<<<..

Consensus 2: >>>>>>>.>.>>>.><........<<<..>>>>>.......<<<<<..<.............<>>>>>.......<<<<<<<<<<<....

Consensus 3: >>>>>>>....>>>>.....>.<....<<<<.>>>>>.......<<<<<.....................><.>>>>>.....><<<<<.<<<<<<<<....

Consensus 4: >>>>>>>...>>..>......<....<<..>>>>>..><...<<<<<.............>>>>>.......<<<<<.<<<<<<<.

Consensus 5: >>>>>>>..>>>>....>.<...<<<<..>>>>>.......<<<<<.....>>>>>.>...<.<<<<<<<<<<<<.

Consensus 6: >>>>>>>.>>>>...>.<......<<<.>>>>>.......<<<<<.....................<>>>>><.........<<<<<<<<<<....

Consensus 7: >>>>>>>.>.>>..><.........<<<.>>>>>.......<<<<<.....................>>>>>.......<<<<<..<<<<<<<....

Consensus 8: >>>>>>>..>>>>.><.......<<<<.>>>>>.......<<<<<..............>>>>>>..<...<<<<<<<<<<<<.

Consensus 9: >>>>>>>..><.><>..........><.>>>>>..><...<<<<<..............<>>>>>>....<<.<<<<<<<<<<.
```

Aligned consensus secondary structure sequences using Center-Star like algorithm:

```
seq[0]: .>>>>>>>-...>>->--.><....-><..<<<>.>>>>>.......-<<<<----<---------.<...---->>>>---->.>..<..<-<<<-<<<<<<<---.

seq[1]: .->>>>>>-.>.>>->.........--....<<<..>>>>>.......-<<<<----<.............--<->>>>---><->.><><-<<<-<<<<<<<--..

seq[2]: ->>>>>>>-.>.>>->--.><....--....<<<..>>>>>.......-<<<<--..<.............--<->>>>--->.......<-<<<-<<<<<<<....

seq[3]: ->>>>>>>....>>->>.....>.<--....<<<<.>>>>>.......<<<<<.....................-><.>>>>--->.....><<-<<<.<<<<<<<....

seq[4]: ->>>>>>>-...>>..>.....-.<--....-<<..>>>>>..><...<<<<<-----.............->-->>>>---.......<<-<<<.-<<<<<<<---.

seq[5]: ->>>>>>>-..>>-->>....>.<--...<<<<..>>>>>.......<<<<<---------------.....>-->>>>---.>...<.<<-<<<--<<<<<<<---.

seq[6]: ->>>>>>>---.>>-->>-...>.<......<<<-.>>>>>.......<<<<<.....................<>-->>>><.........<-<<<--<<<<<<<....

seq[7]: ->>>>>>>-.>.>>..->-<..-.-......<<<-.>>>>>.......<<<<<.....................-...>>>>-->.......<<-<<<..<<<<<<<....
```

```
seq[8]: ->>>>>>>..>->>->.>-<----.......<<<<.>>>>>.......<<<<<--------.............>>>>>-->>..<...<<-<<<-<<<<<<<---.

seq[9]: ->>>>>>>..><--.><>--..........>---<.>>>>>..><...<<<<<-------.............<>>>>-->>-.-...<<.<<<-<<<<<<<--<.
```

APPENDIX F: Combined big multiple alignment after aligning ten multiple alignments from ten Sub-SCFGs

```
aseqs[0]:
.GGGCCCG.UAGCU.U..AGUcuGG..UA..GAGCGCCUGACUUUUAA.UCAG....G.........CGGUC....GAGG...GUUCGAAUC.CCU.UCGGGCCC...G
aseqs[1]:
G.CCCCCA.UCGUC.UAGA.GGCCu.....AGGACAUCUCCCUUUCAC.GGAG....GCAA.............C.GGGG...A.UUCGAAU.UCC.CCUGGGGG..UA
aseqs[2]:
.GUCUUGA.UAGUA.U..AA...AC...A..UUACUCUGGUCUUGUAA.ACCU..GAAA..............U.GAAG...AUCUU..CU.CUU.CUCAAGACA...
aseqs[3]:
.GCCCAUGU..AGC.UCA.GUAGGA...U.AGAGCACGCGCCUUCUAAGCGUGA.................G.GUCGGAA...GUUCGAGCC.UUC.UCGUGGGCACCA
aseqs[4]:
.GGCGGCA.UGGCCAAG.U.GG..U..A..A.GGCAGGGGACUGCAAAUCCUU.....U.........AC.C.C..CCAG...UUCAAAUCU.GGG..UGCCGCC...U
aseqs[5]:
.ACUUUUA..AAGG..AUAAC.AGC..C..AUCCG.UUGGUCUUAGGCCCCAA..............AAA.U.U..UUGG...UGCAACUCC.AAA..UAAAAGU...A
aseqs[6]:
.GGCGGCA...UGG..CC.AAGC.GGUA..AGGC.AGGGGACUGCAAAUCCUUU..............AU.CC..CCAGUUCAAAU...CU.GGG..UGCCGCCU...
aseqs[7]:
.ACCUGGG.UAGUUUA.A.A...G.GUA..AAAC.CUUAAUUUCAUACAUUAAAGA.................U.GAGA..AUUCGAUUUU.CUC..CCCAGGUA...
aseqs[8]:
.UUAAAAGUAG.UU.AAU.A.....U...A.UAACUUAGAAUUGUCAGUUCUA........G..........AU.UCCUU....UU.ACU.A.AGG..CUUUUUA...U
aseqs[9]:
.CUCUCGGUAGC..CAAG..UUGGUUuAAGG...CGCAAGACUGUAAAUCUUG.......AGA...........UCGGGC..GU.U.CGACUCGCC..CCCGGGA..GA
aseqs[10]:
.GCCUUGA.AAGCU.C..AAC..AA..CU.AGAGCUUUGGUCUUGUAA.ACCA....G.........GAG.A....GAGG...GU.AAACUC.CCU.CUCAAGGC...U
aseqs[11]:
G.CUCUCU.UAGCU.UAAU.GGUU......AAAGCAUAAUACUUCUAA.UAUU....AAUA..........U..U.CCAU...G.UUCAAAU.CAU.GGAGAGAG..UA
aseqs[12]:
.GGGGGUA.UGGCG.A..AAUU.GG..UA.GACGCUGCGGACUUAAAA.UCCG..UUGGcuuuaaagaccg...U.GAGG...GUUCAAGUC.CCU.CUACCCCCA...
aseqs[13]:
.GGUCGGAU..GGU.GUA.GUCGGU...U.AUCACGGUUGCUUUACACGCAACA.................G.GUCUCGA...GUUCGAUCC.UCG.GUCGGAUCA...
aseqs[14]:
.AAAAAAG.UAGUUUAA.U.UU..A..A..G.AACGAUAGGUUGUGGGGCUAU.....U.........AG.C.G..GU......GUUA.....ACC..CUUUUUU...A
aseqs[15]:
.GAGAGUA..UUGU..UUAAU.GGU..A..AAACAuUCUGUCUUUUAAGCAGU..............CUA.U.A..GUGG...UUCGAUUCC.ACU..UAUUCUC...A
aseqs[16]:
.GCGGGUA...UAG..UU.UAGU.GGUA..AAAC.CUUAGCCUUCCAAGCUAAC..............GA.UG..CGGGUUCGAUU...CC.CGC..UACCCGCU...
aseqs[17]:
.GGCCUGU.UGGUGAA.G.C...G.GUUA.ACAC.ACACGGUUUUCAUCCGUGGAC.................ACACGG..GUUCGAACCC.CGU..ACAGGCUACCA
```

```
aseqs[18]:
.GGGAAAGU.U.UC.CAU.G......GGUAgGGUAAGAUAUUUGCUAAAUAUU........GugcguuugcaCA.UAGAU..GUUCGAAUCA.UCU..CUUUCCC...G
aseqs[19]:
.GAUUGGGUGGC..AGAU..A.GG.G..AUG...CACUAGAUUGUAAAUCUAG.......GUA............GGAAG..GU.U.CAAGUCCUU..UCCUGGU..CA
aseqs[20]:
.GGGUUAU.UAGCU.C..AGUu.GG..UU.AGAGCACACCCCUGAUAA.GGGU....G.........AGGUC....CCUG...GUUCAAAUC.CAG.GAUAACCC...A
aseqs[21]:
A.AAAAGA.UAAGC.UAAU...UA.......AGCUACUGGGUUCAUAC.CCCA....UUUA.............U.AAAG...G.UUAUAAU.CCU.UUUCUUUU..UA
aseqs[22]:
.UCCGUGA.UAGUU.U..AAU..GG..UC.AGAAUGGGCGCUUGUCGC.GUGC..CAGA..............U.CGGG...GUUCAAUUC.CCC.GUCGCGGAG...
aseqs[23]:
.GGAGAGAU..GGC.UGA.GU.GGA...CuAAAGCGGCGGAUUGCUAAUCCGUUGUACGAGUU.A.Aucg.U.ACCGAGG...GUUCGAAUC.CCU.CUCUUUCCG...
aseqs[24]:
.GCCCGGC.UAGCUCAG.UCGG..U..A..G.AGCAUGAGACUCUUAAUCUCA.....G.........GGUC.G..UGGG...UUCGAGCCC.CAC..GUUGGGC...G
aseqs[25]:
.GGAAAUG..UGCC..CGAAA......GU.AGGGA.UCACUUUGAUAGAGUGA..............AAUAU.A..UGGG...UUCAAACCC.CAU..CAUCUCC...U
aseqs[26]:
.GAAACUA...UAA..UU.CAAUuGGUU.AGAAU.AGUAUUUUGAUAAGGUACC..............AA.UA..UAGGUUCGAUU...CC.UGU..UAGUUUCA...
aseqs[27]:
.GGUUUCG.UGGUCUA.G.UC..G.GUUA.UGGC.AUCUGCUUAACACGCAGAACG..................UCCCCA..GUUCGAUCCU.GGG..CGAAAUCA...
aseqs[28]:
.GUGGUUCUAG.UU.UAG.U......GA.A.AAACGUUUGCUUUGCAAGCAGA........A..........AU.CCUAA..GUUAAAUUCU.UAG..GAACUAC...A
aseqs[29]:
.CCGACCUUAGC..UCUG..UUGGUA..UAG...CGGAGGACUGUAGAUCCUU.......AGG...........UCACUG..GU.U.CGAAUUCGG..UAGGUCG..GA
aseqs[30]:
gGCGGACG.UAGCC.A..AGU..GG..AUuAAGGCAGUGGAUUGUGGA.UCCU....C.........UACGC....GCGG...GUUCAAUUC.CCG.UCGUUCGC...C
aseqs[31]:
G.UAAAUA.UAGUU.UAA....CC......AAAACAUCAGAUUGUGAA.UCUG....ACAA.............C.AGAG...G.CUUACGA.CCC.CUUAUUUA..CC
aseqs[32]:
.ACUCUAA.UAGUU.U..AU...GA...A..AAACAUUGGUCUUGUAA.ACCA..AAAA.............C..U.GAAG...ACUCCACC..CUU.CUUAGAGUA...
aseqs[33]:
.GGGCUAUU..AGC.UCA.GUUGGU...U.AGAGCGUUGCUUUGAUAAGGCAAA.................A.GUCGAAA...GUUCAAAUC.UUU.CAUAGCCCA...
aseqs[34]:
.GGCGCGA.UGGCAGAG.U.GG..U..CuaA.UGCGUGAGACUUGAAAUCUCA.....UuucuucggaAG.C.G..UCGG...UUCAAAUCC.GGC..UCGCGUC...G
aseqs[35]:
.AGCUUCU..UAAC..UCAGG.GGU..A..GAGUG.CGAGGCCCAUAACCUCG.............AGGUC.C..UAGG...AUCGAAACC.UAG..AGAAGCU...A
aseqs[36]:
.GUCUUUG...UAG..UA.CAU...CUA..AUAU.ACUGGUCUUGUAAACCAGA..............GA.AG..GAGAA.CAACU.aaCC.UCC..CUAAGACU...
aseqs[37]:
.GGGCCCA.UAGCUCA.G.U...G.GUA..GAGU.GCCUCCUUUGCAAGGAGGAUG.................CCCUGG..GUUCGAAUCC.CAG..UGGGUCCA...
aseqs[38]:
.AGAGAUUUAA.GU.UAA........UA.A.AACUGAAAACCUUCAAAGUUUU........A..........AA.UAAGA..GUGGAACUCU.CUU..AGUCUUU...A
```

```
aseqs[39]:
.GCGUCCAUUGU..CUAA..U.GGAU.AGGA...CAGAGGUCUUCUAAACCUU.......UGG............UAUAG..GU.U.CAAAUCCUA..UUGGACG..CU
aseqs[40]:
.GUAAAUA.UAGUU.U..AAC......CA..AAACAUUAGAUUGUGAA.UCUA....A.........UAA.U....AGGG...CCCCACAAC.CCC.UUAUUUAC...C
aseqs[41]:
G.GGGACG.UAGCU.CAAUuGGUA.......GAGCGUAUGUUUUGCAA.GCAU....AAAG..........C..U.GUCG...G.UUCAAAU.CCG.AUCGUCUC..CA
aseqs[42]:
.GUCUCUG.UGGCG.C..AAUC.GG..UU.AGCGCGUUCGGCUGUUAA.CCGA..AAGA............U..U.GGUG...GUUCGAGCC.CAC.CCAGGGACG...
aseqs[43]:
.CGGGAAGU..GGC.UCA.GUUUGG...U.AGAGCAUUCGGUUUGGGACCGAAG.................G.GUCGCAG...GUUCAAAUC.CUG.UCUUCCCGACCA
aseqs[44]:
.GGGCCCG.UCGUCUAGcCUGG.uU..A..G.GACGCUGCCCUGACGCGGCAG.....A.........AAUC.C..UGGG...UUCAAGUCC.CAG..CGGGCCC...A
aseqs[45]:
.GCGGAAG..UAGU..UCAGU.GGU..A..GAACA.UCACCUUGCCAUGGUGG..............GGGUC.G..CGGG...UUCGAAUCC.CGU..CUUCCGC...U
aseqs[46]:
.GCGGGUA...UAG..UU.UAGU.GGUA..AAAC.CCUAGCCUUCCAAGCUAAC...............GA.UG..CGGGUUCGAUU...CC.CGC..UACCCGCU...
aseqs[47]:
.CGGCACG.UAGCGCA.G.CCu.G.GUA..GCGC.ACCGUCAUGGGGUGUCGGGGG.................UCGGAG..GUUCAAAUCC.UCU..CGUGCCGACCA
aseqs[48]:
.GUUAAUGUAG.CU.UUA.A......AAUUaAAGCAAGGCACUGAAAAUGCCU........A..........GA.UGAGU..GCUCCAA....CUC..CAUAAAC...A
aseqs[49]:
.UGGGUUGUAGC..CUAA..U.GGAA..AGG...CAAUUGGCCGUUAACCAGG.......AGA...........UAACAA..GA.U.CAAUACUUG..UCAACUC..AG
aseqs[50]:
.GCGGAUA.UAGAU.U..AAA..GG..UA..AAUUAUCUGCCUUCCAA.GCAG....A.........GGA.U....AUGG...GUUCGAUUC.CCG.UUAUCCGC...A
aseqs[51]:
G.GUCUUA.UAGUC.AAU....A.......AUGAUAUCAAACUGCAAU.UUUG....AAGG.............A.GUAA......GUU.....UU.ACUAAGGC..UU
aseqs[52]:
.GGCCUUG.GGGUG.U..C.........A..ACACGUGGGGUUGCAAA.CCCC..AAGA...............U.GCAG...UAUAAUAC..CUG.CCGGGGCUU...
aseqs[53]:
.GGCCCCAU..AGC.GAAcGUUGGU...U.AUCGCGCCUCCCUGUCACGGAGGA.................G.AUCACGG...GUUCGAGUC.CCG.UUGGGGUCGCCA
aseqs[54]:
.GGGGGUA.UAGCUCAG.UUGG..U..A..G.AGCGCUGCCUUUGCAAGGCAG.....A.........UGUC.A..GCGG...UUCGAGUCC.GCU..UAUCUCC...A
aseqs[55]:
.GCUCGAU..UAGC..UCAGCuGGU..UA.GAGCA.UGCGGCUGUUAACCGCA..............AGGUC.G..UAGG...UUCGAUCCC.UAC..AUCGAGC...G
aseqs[56]:
.GUUCAUG...UAG..CU.UAAA.ACCA..AAGC.AAGGCAUUGAAAAUGCCUA..............GA.UG..AGUA.U..AUU...AA.CUC..CAUAAACA...
aseqs[57]:
.GGCCCCA.UAGCGAA.G.UU..G.GUUA.UCGC.GCCUCCCUGUCACGGAGGAGA.................UCACGG..GUUCGAGUCC.CGU..UGGGGUCGCCA
aseqs[58]:
.GCCGAAAUAG.CU.CAG.U....U.GGGA.GAGCGUUAGACUGAAGAUCUAA........A..........GGUCCCCG..GUUCAAUCCC.GGG..UUUCGGC...A
aseqs[59]:
.AAGAGCUUAAG..UUAA..A....U.AAAC...UGAAAGCCUUCAAAGCUUU.......UUA............UAAGA..AU.G.GAAACUUCU..UAGCUCU..UG
```

```
aseqs[60]:
.GAUACGG.UAGCU.U..AAU......UA..AAGCGUCUCAUUGAAAA.UGAG....G.........AAGAU....GGUA....CUUUAG...UAC.CUUGUGUC...A
aseqs[61]:
G.GCCGAA.UGGUC.UAGU.GGUA.......UGAUUCUCGCUUUGGGU.GCGA....GAGG..........U..C.CCGG...G.UUCAAUC.CCC.GGUUCGGC..CC
aseqs[62]:
.AAGAGUA.UAGUU.U..AAA..GG..UA..AAACAGAAAGCUUCAAC.CUUU..AAUU..............U.CUUA...GUUCGAGUC.UAA.GUGCUCUUG...
aseqs[63]:
.GGGGGUAU..AGC.UCA.GU.UGG...U.AGAGCGCUGCCUUUGCACGGCAGA................U.GUCAGGG...GUUCGAGUC.CCC.UUACCUCCA...
aseqs[64]:
.GGGGCCU.UAGCUCAG.CUGG..G..A..G.AGCGCCUGCUUUGCACGCAGG.....A.........GGUC.A..GCGG...UUCGAUCCC.GCU..AGGCUCC...A
aseqs[65]:
.UGGAGUA..UAGC..CAAGU.GGU..A..AGGCA.UCGGUUUUUGGUAUCGG..............CAUGC.A..AAGG...UUCGAAUCC.UUU..UACUCCA...G
aseqs[66]:
.ACUUUUA...AAG..GA.UAAC.AGCU..AUCC.AUUGGCCUUAGGAGUCAAA..............AA.UA..UUGGUGCAACU...CC.AAA..UAAAAGUA...
aseqs[67]:
.GGGGAUG.UGGUGGA.A.UU..G.GUAG.ACGC.AACGGACUUAAAAUCCGUCGAUUGUAUAGAUc.......GUGAGG..GUUCAAGUCC.CUC..CGUCCCCA...
aseqs[68]:
.AGCGGGGUAG.GC.CAG.C....CaGGUA.GGCCGCGGGGCUCAUAACCCCG........A.........GGUCCCGG..GUUCAAAUCC.CGG..CCCCGCU...A
aseqs[69]:
.GGGAGAGUGGC..CGAG..C.GGUCaAAAG...CGACAGACUGUAAAUCUGU.......UGAagguuuucua..CGUAG..GU.U.CGAAUCCUG..CCUCUCC..CA
aseqs[70]:
.UCCGCAG.UAGCU.C..AGU..GG..UA..GAGCUAUCGGCUGUUAA.CCGA....U.........CGGUC....GUAG...GUUCGAAUC.CUA.CCUGCGGA...G
aseqs[71]:
G.CGUCCA.UCGUC.UAAA.GGAU......AGGACAGAGGUUUUCUAA.ACCU....CCAG.............U.AUAG...G.UUCGAAU.CCU.AUUGGACG..UA
aseqs[72]:
.CGCGGGG.UGGAG.C..AGCCuGG..UA..GCUCGUCGGGCUCAUAA.CCCG..AAGA............U..C.GUCG...GUUCAAAUC.CGG.CCCCCGCAAcca
aseqs[73]:
.GGAAAGAU..GGU.UGA.GU.GGU...UuAAGGCGUAGCAUUGGAAAUGCUAUGUAGGCUUU.U.Gguc.U.AUCGAGG...GUUCGAAUC.CCU.CUCUUUCCG...
aseqs[74]:
.AGGGCUA.UAGCUCAG.C.GG..U..A..G.AGCGCCUCGUUUACACCGAGA.....A.........UGUC.U..ACGG...UUCAAAUCC.GUA..UAGCCCU...A
aseqs[75]:
.UGGGGCG..UGGC..CAAGU.GGU..A..AGGCA.ACGGGUUUUGGUCCCGC..............UAUUC.G..GAGG...UUCGAAUCC.UUC..CGUCCCA...G
aseqs[76]:
.GGAGGUA...UGG..CU.GAGU.GGCUuAAGGC.AUUGGUUUGCUAAAUCGACauacaauaagauuguAU.CA..UGGGUUCGAAU...CC.CAU..UUCCUCCG...
aseqs[77]:
.GGAGAGU.UGGCAGA.G.C...G.GUA..AUGC.AGCGGACUCGAAAUCCGCCGAGCCAAUGUUGaauugg.gUCGCAG..GUUCAAAUCC.UGU..ACUCUCCU...
aseqs[78]:
.GCCCUUUUAA.CU.CAG.U......GGUA.GAGUAACGCCAUGGUAAGGCGU........A.........AGUCAUCG..GUUCAAAUCC.GAU..AAGGGGC...U
aseqs[79]:
.AUAGAUAUAAG..UUAA..GUGGUA..AAC...UGAAUGUCUUCCACACAUU.......GAU............UGUGA..GU.U.CGAUUCUCA..CUAUCUA..GA
aseqs[80]:
.GCAUAUA.UAGUA.A..ACA..AU..A...UUAUAUUUGCCUUCCAA.GCAA....A.........AGU.C....CUAA.....UAAAA...UUA.GUGUAUGC...U
```

```
aseqs[81]:
G.UGGACG.UGCCG.GAGU.GGUU......AUCGGGCAUAACUAGAAA.UCAU....GUGGgcuuugcccg...C.GCAG...G.UUCGAAU.CCU.GCCGUUCA..CG
aseqs[82]:
.UCCUCGG.UAGCU.C..AAUU.GG..CA..GAGCAGCCGGCUGUUAA.CCGG..CAGG............U..U.ACUG...GUUCGAGUC.CAG.UCCGGGGAG...
aseqs[83]:
.GGUCCUAU..AGU.GUA.GU.GGU...U.AUCACUUCGGACUUUGAAUCCGAA.................A.A.CCCAG...GUUCGAAUC.CUG.GUAGGACCA...
aseqs[84]:
.GUCUACG.UAGCUUAA.CCCC..C..A..A.AGCAAGACACUGAAAAUGCCU.....A.........GA.U.G..GAU....UCACAC..A.UCC..CAUAGAC...A
aseqs[85]:
.GGGAUUG..UAGU..UCAAUuGGU..UA.GAGUA.CCGCCCUGUCAAGACGG..............AAGUU.G..CGGG...UUCGAGCCC.CGU..CAAUCCC...G
aseqs[86]:
.GGUGAGG...UGU..CC.GAGU.GGCUgAAGGA.GCACGCCUGGAAAGUGUGUauacggcaacg....UAuCG..GGGGUUCGAAU...CC.CCC..CCUCACCGcca
aseqs[87]:
.AGAGAGU.UGGCUGA.G.U...G.GUA..AGGC.GACUAGCUUGAGUCUAGUUAAGUUAAAACUU........UCAUAU..GUUCGAAUCA.UAU..ACUCUCUG...
aseqs[88]:
.CGGGAGAGAA.UU.UAA........AUUA.GAAUGUUGGCUUUGGGGGUCAA........U..........AG.UGGAG..GUUUGAGUCC.UUC..UUUCUCG...A
aseqs[89]:
.AAAUAUGAAGC..GAUU..U....A..UUG...CAAUUAGUUUCGACCUAAU.......CU.............UAGGU..GA.A...AUUCACC..CCAUAUU..UU
aseqs[90]:
.CAAGGAA.UAGUU.U..AUG......UA..GAAUUCCAGCUUUGGGU.GUUG....G.........UGG.U....GAGG...UUU..AAUG.UCU.CUUCCUUG...A
aseqs[91]:
U.AGAAUG.AAGCU.CGCU.GGAU......AGAGUGUUUAGCUGUUAA.CUAA....AAUA..........U..U.ACGG...G.AUCGAGG.CCC.GUCAUUCU..AG
aseqs[92]:
.GCCCCCA.UCGUC.U..AGU..GG..UUcAGGACAUCUCUCUUUCAA.GGAG..GCAG..............C.GGGG...AUUCGACUU.CCC.CUGGGGGUA...
aseqs[93]:
.GCCCGGAUgaACC.AUG.GC.GGU...C.UGUGGUGCAGACUUCAAAUCUGUAGGCGGUUAG.C.G....C.CGCAGUG...GUUCGACUC.CACcUUUCGGGUG...
aseqs[94]:
.ACUUUCU.UAGUAUUA.A.CC.....A..G.UACACGUGACUUCCAAUCACA.....A.........AG.U.C..UUAG...UAGAA.UCU.AAG..AGAAAGU...A
aseqs[95]:
.GCAGCAA..GGGU..G..GU......CU..CAAC.CUGGGUUCAUUCCCCAG..............CUA.A.U..AAAG...UUCGAUUCU.UUG..UAGCGGC...U
aseqs[96]:
.AUUCCUG...UAG..UU.GAA....A.C..AAC.AAUAACUUUUCAUGUUAUA..............GG.UU..UAGGUUGAACC...CC.UAA..CAGGAAUC...
aseqs[97]:
.GCGCUCU.UAGUUCA.G.UUu.G.GUA..GAAC.GCGGGUCUCCAAAACCCGAUG.................UCGUAG..GUUCAAAUCC.UAC..AGAGCGUG...
aseqs[98]:
.GCGGAUUUAG.CU.CAG.U....U.GGGA.GAGCGCCAGACUGAAGAUCUGG........A..........GGUCCUGU..GUUCGAUCCA.CAG..AAUUCGC...A
aseqs[99]:
.GGCUCGUUGGU..CUAG..G.GGUA..UGA...UUCUCGCUUCGGGUGCGAG.......AGG...........UCCCGG..GU.U.CAAAUCCCG..GACGAGC..CC
```

APPENDIX G: The alignment/parsing scores for the 100 test data on the SCFGs constructed with different number of training data

	1400	1300	1200	1100	1000	900	800	700	600	500	400	300	200	100
DK0660	83.56	84.24	83.25	85.29	82.65	83.99	83.90	83.56	81.91	70.86	79.23	81.56	76.86	74.05
DE2520	57.71	60.94	60.67	59.80	60.90	60.63	59.27	58.05	58.25	66.02	52.66	51.54	49.19	51.68
DT5320	30.76	34.83	37.14	32.41	33.45	37.26	31.47	34.20	36.37	25.64	29.96	25.34	24.53	31.27
DR1181	66.87	71.50	72.00	72.03	72.83	73.09	72.26	72.42	71.36	58.65	70.67	71.48	77.70	64.50
DC2720	68.59	64.09	65.94	65.71	67.05	65.66	67.43	66.26	65.78	66.21	64.64	61.87	62.86	62.64
DL5720	42.93	41.79	38.70	38.37	41.59	34.83	33.91	31.32	33.12	28.59	35.15	35.60	22.95	37.50
DC2680	70.03	66.17	66.45	67.04	67.06	66.39	67.14	66.07	64.65	67.87	63.91	62.15	64.00	63.21
DM3920	63.03	63.24	62.97	61.48	61.26	61.02	60.29	59.44	60.08	57.63	58.25	56.86	57.22	58.45
DD4700	18.39	13.84	13.06	9.51	8.86	6.64	6.76	7.12	8.47	30.76	7.57	6.46	7.95	22.54
DY6280	71.27	72.68	73.21	73.33	72.77	73.55	73.96	72.72	71.63	63.73	70.17	69.85	63.24	61.51
DT5080	40.11	49.88	43.67	46.66	46.39	45.45	45.26	43.69	45.65	37.65	39.11	41.32	35.87	45.08
DR4000	67.93	69.74	69.01	68.73	68.37	68.86	69.71	68.55	68.10	61.86	71.66	73.05	71.34	59.37
DL2601	62.16	62.96	57.88	58.15	62.94	65.22	63.18	57.50	65.09	46.48	63.12	57.48	64.94	50.05
DV6161	60.71	61.66	61.62	60.91	60.94	61.18	62.04	62.63	62.55	51.91	61.26	59.18	54.42	57.31
DH4700	24.31	26.80	28.11	22.26	29.11	21.39	20.00	17.67	19.23	25.55	20.43	16.46	15.31	25.00
DK4030	56.84	57.84	56.90	56.17	56.90	56.98	56.03	55.66	56.13	47.74	54.00	52.30	52.06	52.63
DG2601	79.91	81.13	80.87	80.70	80.90	78.98	79.11	79.44	78.93	77.41	77.81	76.40	77.74	76.04
DE1140	65.48	64.47	63.92	64.43	63.70	61.92	63.03	62.78	60.69	62.10	57.88	57.12	58.48	52.47
DS3881	23.81	21.71	24.27	20.02	25.88	20.91	21.89	19.01	23.51	13.49	20.73	25.33	22.84	36.71
DY5000	45.64	46.28	44.06	44.93	45.29	43.99	43.84	40.62	42.48	42.47	42.63	38.30	36.92	42.45
DI2580	87.33	87.30	87.49	87.26	87.70	88.04	88.32	87.07	86.03	74.89	85.48	86.74	85.96	74.66
DX4880	48.70	52.43	52.39	51.09	51.23	51.34	50.15	48.60	50.32	32.22	49.97	49.66	43.84	40.85
DD6280	56.18	58.35	57.17	56.68	55.59	55.52	55.23	54.70	54.83	60.62	53.94	52.19	53.10	51.56
DS2920	60.31	60.82	54.33	53.24	57.52	59.61	54.51	49.73	56.85	36.17	53.58	47.63	52.13	51.96
DK7680	74.20	74.86	74.61	73.24	74.09	74.47	74.50	73.69	72.63	59.97	73.62	75.36	76.86	65.73

	1400	1300	1200	1100	1000	900	800	700	600	500	400	300	200	100
DI5120	47.32	41.26	64.98	39.89	39.08	33.35	36.73	35.78	35.18	29.61	31.46	33.33	27.40	42.50
DI4080	64.69	65.03	59.62	64.04	64.14	63.53	63.78	63.59	64.72	57.08	65.85	64.51	61.03	59.25
DV6280	58.16	59.69	51.06	59.62	60.66	59.45	60.62	61.14	63.01	47.88	59.41	57.05	52.43	54.34
DA5040	61.04	52.41	62.08	50.92	48.39	47.56	47.80	48.66	49.51	63.94	50.56	45.68	50.15	55.79
DY6742	62.03	61.54	61.53	62.63	62.20	62.28	60.41	57.52	57.94	50.66	58.70	60.51	57.22	57.27
DH2600	57.83	61.64	44.38	62.02	63.03	62.58	62.31	61.83	62.64	42.15	55.50	51.77	44.37	47.91
DH5880	39.88	46.22	36.38	42.68	44.10	42.26	43.32	41.36	45.34	40.65	41.37	42.14	26.08	38.17
DT5220	40.26	38.54	84.71	35.97	36.59	35.85	36.85	33.38	37.17	34.36	33.65	33.92	35.00	38.90
DI2440	83.07	84.19	56.45	84.68	84.78	85.01	85.21	84.29	83.48	74.42	83.00	83.77	81.25	74.25
DS6241	62.90	58.69	69.30	58.92	57.00	55.23	60.58	58.19	59.28	36.81	62.88	56.58	56.58	49.19
DM6160	68.47	69.07	25.81	68.97	69.20	70.18	69.05	69.08	67.94	45.48	67.51	67.49	63.87	60.48
DT5360	28.47	30.21	79.79	27.01	31.45	25.18	25.89	25.22	26.73	18.20	24.47	30.41	17.68	29.96
DA0380	79.35	80.11	42.30	80.34	80.62	80.55	81.13	80.42	80.10	82.99	79.52	78.31	77.65	69.30
DW4980	42.86	44.48	67.14	40.63	44.33	40.62	39.09	37.81	40.12	40.63	41.14	41.17	38.89	43.04
DR2700	66.48	67.27	44.57	67.03	66.60	66.53	66.59	66.79	66.61	58.05	66.65	66.94	62.20	64.61
DH5840	36.94	43.97	78.97	41.81	44.91	43.05	43.03	41.44	45.40	41.54	44.19	40.61	27.61	36.03
DA4500	76.17	78.97	75.98	79.34	79.69	79.48	80.08	79.97	80.34	76.70	79.43	78.80	76.73	68.70
DQ9991	74.21	76.18	81.80	74.76	74.85	74.98	75.89	75.44	74.47	59.99	75.83	76.52	75.67	66.18
DP1140	82.05	82.47	69.09	81.37	79.95	76.78	81.73	80.28	80.22	56.48	78.82	81.98	75.81	68.05
DV0860	68.41	69.99	82.40	70.01	67.63	67.75	68.84	67.19	65.77	61.44	65.46	65.44	62.89	65.20
DG1500	81.43	82.11	76.55	82.69	82.58	82.24	82.44	82.27	82.21	77.93	80.91	79.60	82.18	71.65
DG2681	75.59	76.85	63.86	76.57	76.88	74.52	74.66	75.12	74.82	71.86	73.75	72.61	74.51	73.08
DP1662	64.62	63.39	34.55	65.82	64.43	64.29	64.61	64.95	63.29	46.88	64.89	67.25	64.26	56.94
DF5920	37.20	36.69	51.22	30.52	29.07	32.69	33.86	31.54	31.36	39.27	26.59	27.56	24.63	36.66
DN5000	49.22	51.15	73.38	50.65	50.16	50.56	50.78	51.04	50.07	40.43	49.58	48.95	41.95	49.10
DG4501	73.16	73.68	17.03	72.63	72.68	72.59	72.32	72.51	73.25	74.11	73.28	72.70	72.78	69.00
DC4840	16.56	16.20	16.54	11.58	14.65	14.30	9.94	7.40	10.14	31.35	8.41	5.48	6.12	17.92
DC5160	17.87	19.79	70.43	15.13	15.55	15.04	16.42	14.30	18.33	27.76	14.08	10.88	14.12	25.67
DD1180	69.64	69.47	84.60	69.84	70.64	70.55	70.52	69.95	69.83	73.74	68.41	68.53	62.60	63.19
DA2520	83.06	84.06	83.06	85.02	85.41	84.86	85.24	84.95	85.12	85.75	84.76	82.37	82.53	71.14

	1400	1300	1200	1100	1000	900	800	700	600	500	400	300	200	100
DN6060	83.89	83.25	77.55	82.54	82.66	82.99	82.05	81.43	80.83	69.47	80.06	81.72	80.63	72.31
DF5970	32.56	37.07	81.89	35.35	37.43	36.12	38.60	37.68	42.17	49.67	37.32	36.33	31.00	36.55
DD1140	76.65	77.12	40.94	77.56	77.53	77.27	77.85	77.18	76.20	80.24	75.50	75.48	71.23	68.68
DF7740	82.61	82.54	35.56	81.38	81.96	81.68	80.96	80.38	80.00	81.95	79.94	78.80	81.16	73.71
DW5040	38.15	40.48	70.36	40.07	39.84	39.38	37.44	36.71	37.74	36.57	41.17	37.26	40.50	34.61
DF4700	33.73	36.47	53.50	34.14	34.95	33.24	36.01	31.34	32.81	38.81	32.29	33.15	26.86	32.51
DP7560	69.90	70.64	86.95	70.51	70.45	69.60	70.49	70.72	69.76	56.34	67.48	63.68	59.27	61.80
DW3960	55.20	53.76	80.72	52.44	51.53	51.24	50.41	49.77	48.54	53.68	49.27	47.66	48.55	51.72
DA2540	85.57	86.39	69.18	87.30	87.58	87.08	87.31	87.23	87.24	86.96	86.62	84.67	83.91	73.74
DA1540	78.19	80.25	41.26	81.22	81.44	80.42	80.28	79.94	80.15	83.65	79.58	77.72	78.59	71.00
DQ4440	68.13	68.47	59.57	69.84	68.92	68.33	68.64	69.34	68.01	55.41	63.07	61.48	59.57	58.06
DL5560	45.74	44.00	77.22	40.86	44.11	37.27	37.48	35.37	37.01	28.37	37.17	37.95	25.43	39.37
DL2180	61.83	64.96	57.35	54.81	64.71	61.79	59.80	54.91	62.71	42.77	61.80	58.21	62.65	51.52
DX0980	80.06	76.60	84.90	78.29	76.89	77.17	76.89	77.23	74.37	53.41	74.15	74.83	70.63	69.30
DY4480	56.76	61.38	64.37	55.90	59.97	62.37	57.03	52.98	58.94	34.16	50.55	50.74	52.62	49.14
DN1541	84.58	84.80	68.82	84.60	84.41	84.46	84.26	83.92	82.81	69.01	80.91	80.83	81.74	73.47
DR2602	63.09	63.46	52.19	63.51	64.19	63.79	63.50	63.06	63.32	54.79	63.61	63.28	60.06	62.60
DX1660	70.23	68.54	75.01	69.95	67.80	68.67	69.38	70.30	67.08	46.77	67.85	69.80	62.80	59.65
DS2602	57.95	54.13	64.61	49.26	52.70	50.85	48.19	45.84	52.20	34.51	49.36	50.99	47.57	51.62
DV2601	74.59	74.78	47.72	75.51	74.99	74.48	73.94	74.30	74.46	64.55	74.34	75.65	71.69	62.47
DQ2920	66.51	63.67	55.73	65.73	65.28	64.93	65.75	65.81	63.86	53.21	64.94	64.60	64.64	57.53
DS4362	48.98	55.39	73.42	46.95	51.14	47.63	41.84	36.38	44.74	24.74	45.75	41.20	37.87	43.63
DS1520	60.03	64.03	52.99	55.29	59.83	60.35	49.59	50.47	55.88	32.47	51.97	53.03	58.95	52.19
DT3360	71.18	72.89	36.37	73.66	72.77	72.74	72.75	72.88	73.43	64.64	74.71	76.31	72.85	62.51
DG4080	51.68	52.83	44.26	52.75	52.83	52.86	51.07	52.08	52.45	47.30	52.79	51.31	46.48	51.43
DG4700	36.43	38.12	76.93	34.46	33.32	33.66	33.94	33.82	34.92	34.46	36.34	33.27	32.96	36.54
DS6744	54.45	44.59	80.26	50.80	52.25	53.09	52.91	53.45	46.88	23.05	43.67	45.10	46.48	41.50
DN1350	76.46	77.48	32.08	76.64	76.66	76.53	76.49	76.13	74.87	61.40	73.88	74.31	73.33	64.16
DQ6050	78.79	80.66	71.06	80.60	80.13	80.01	81.04	81.17	80.79	73.76	75.41	72.81	67.71	69.26
DF5400	22.22	31.69	46.15	30.78	31.48	33.09	34.08	33.97	35.06	34.28	33.97	33.78	26.69	34.86

	1400	1300	1200	1100	1000	900	800	700	600	500	400	300	200	100
DD2600	68.13	71.19	40.95	71.00	71.51	71.39	71.87	72.11	72.69	68.95	71.92	72.72	72.40	62.33
DS1664	52.13	50.51	8.24	46.70	47.39	50.65	44.14	42.98	45.85	25.05	42.45	38.12	40.23	41.17
DS3880	49.78	52.98	29.09	46.38	50.30	49.86	45.16	38.99	46.33	28.11	45.28	45.29	42.26	42.54
DP5120	40.72	42.04	57.18	40.35	39.77	37.09	36.27	37.37	36.48	30.89	38.39	35.83	36.46	43.28
DR4800	5.89	4.05	61.62	3.95	6.80	6.00	2.53	1.62	3.26	6.66	3.45	2.39	5.57	9.98
DP5200	33.61	29.67	9.53	27.13	29.09	26.25	24.68	23.67	26.60	26.12	24.24	25.83	26.85	33.54
DN5160	52.15	56.78	28.84	56.87	56.52	56.71	57.45	56.78	56.62	39.06	54.91	56.09	46.50	45.73
DE2500	59.12	61.50	6.71	62.09	61.24	61.68	62.96	61.26	61.42	64.43	58.64	54.98	56.07	51.86
DZ7560	21.57	9.76	39.78	5.70	8.68	7.44	8.37	6.88	10.13	4.60	5.75	11.40	7.79	24.52
DG5120	30.33	33.69	79.09	29.99	30.18	29.59	29.16	26.80	29.25	38.88	29.59	31.45	32.55	35.10
DX3720	7.84	6.44	76.06	3.20	7.50	6.23	7.68	8.52	8.25	-6.08	-0.10	-3.40	-0.51	20.93
DE4980	37.21	41.38	65.86	38.05	38.28	36.08	40.11	37.71	41.59	49.34	40.16	39.26	37.44	34.37
DW2720	79.61	80.17	83.25	78.59	76.73	73.92	78.40	76.42	77.02	71.36	76.22	79.07	74.70	66.88
DF6280	76.84	77.12	60.67	75.43	76.16	76.35	75.69	75.18	74.37	77.37	74.77	74.15	76.42	69.66
DP8041	65.39	66.04	37.14	65.92	66.25	65.92	66.72	67.24	66.02	48.42	63.36	60.07	54.74	57.45

Note: the first row gives the number of training data to construct a SCFG; the first column gives the test data code (Eddy 1994); the remaining entries contain *the negative log likelihood* (*NLL*) score of the sequence; the *NLL* score = -log(Prob$(s \mid G)$), the probability of a sequence given by the grammar *G*.

APPENDIX H: The alignment/parsing scores for the 100 test data on the CMs constructed with COVE

Note: the first row gives the number of training data to construct a CM; the first column gives the test data code.

	1400	1300	1200	1100	1000	900	800	700	600	500	400	300	200	100
DK0660	83.25	84.20	83.23	84.61	85.34	84.16	83.59	83.57	81.04	70.86	80.67	81.50	76.86	75.04
DE2520	58.22	59.58	60.66	59.76	59.75	60.69	59.06	58.05	58.46	66.02	53.39	51.49	49.19	54.41
DT5320	33.59	33.34	37.15	32.64	31.23	37.38	30.77	34.20	36.49	25.64	30.31	25.29	24.53	37.87
DR1181	72.41	71.29	72.01	72.12	72.79	73.17	72.45	72.42	71.61	58.65	72.57	71.48	77.70	66.28
DC2720	65.07	64.64	65.94	65.78	65.63	65.89	67.24	66.25	66.16	66.21	64.38	61.86	62.86	61.55
DL5720	40.44	41.02	38.70	38.48	41.54	34.92	34.10	31.32	33.26	28.59	34.73	35.55	22.95	39.38
DC2680	65.62	66.16	66.45	67.08	67.04	66.59	67.00	66.06	65.65	67.87	64.94	62.15	64.00	62.13
DM3920	64.36	63.24	62.97	61.57	60.96	61.10	59.82	59.44	60.22	57.63	59.93	56.87	57.22	59.00
DD4700	14.89	14.24	13.65	9.72	8.46	7.05	6.99	7.11	8.48	30.76	8.95	6.46	7.95	23.66
DY6280	72.82	72.41	73.22	73.31	72.75	73.64	73.85	72.72	72.06	63.73	70.08	69.84	63.24	64.98
DT5080	46.90	49.85	43.68	46.71	47.51	45.45	44.39	43.68	45.21	37.65	38.50	41.31	35.87	46.93
DR4000	69.64	69.48	69.01	68.84	68.51	68.90	69.55	68.55	68.58	61.86	72.27	73.06	71.34	65.08
DL2601	60.55	59.76	57.89	58.20	60.60	66.39	63.06	57.50	61.84	46.48	63.30	57.47	64.94	51.37
DV6161	60.65	61.54	61.60	60.88	60.58	61.35	61.92	62.63	62.70	51.91	61.56	59.18	54.42	58.17
DH4700	24.34	26.77	28.11	22.04	27.37	21.47	18.68	17.67	19.45	25.55	17.60	16.41	15.31	27.04
DK4030	57.62	57.65	56.91	56.24	56.38	57.00	55.86	55.66	56.32	47.74	53.93	52.31	52.06	54.59
DG2601	79.96	81.05	80.88	80.72	80.91	79.00	78.96	79.42	79.10	77.41	77.58	76.40	77.74	76.59
DE1140	63.90	64.21	63.91	64.48	63.58	62.65	62.88	62.78	61.20	62.10	59.58	57.13	58.48	57.31
DS3881	26.73	22.99	24.25	20.23	24.94	22.66	21.74	19.02	28.20	13.49	21.46	25.34	22.84	30.09
DY5000	48.88	46.65	44.07	45.28	45.95	43.75	43.74	40.62	41.74	42.47	42.12	38.25	36.92	43.12
DI2580	87.43	87.06	87.51	87.27	87.48	88.10	88.21	87.07	86.37	74.89	85.75	86.74	85.96	76.48
DX4880	51.19	52.82	52.40	50.98	51.05	51.37	48.63	48.60	50.47	32.22	51.07	49.67	43.84	48.61
DD6280	57.96	58.20	57.16	56.72	55.55	55.49	55.12	54.71	55.04	60.62	54.92	52.13	53.10	54.90
DS2920	56.45	55.44	54.32	53.09	56.00	60.69	54.49	49.74	56.12	36.17	54.81	47.63	52.13	49.23

	1400	1300	1200	1100	1000	900	800	700	600	500	400	300	200	100
DK7680	74.03	74.66	74.60	73.25	73.89	74.66	74.35	73.69	72.74	59.97	73.23	75.36	76.86	66.12
DI5120	40.35	41.14	40.87	38.98	39.39	33.56	35.83	35.77	35.25	29.61	32.63	33.33	27.40	45.87
DI4080	65.14	64.95	64.99	64.12	63.81	63.60	63.62	63.59	64.88	57.08	66.32	64.52	61.03	60.12
DV6280	60.19	59.30	59.60	59.34	60.46	59.64	60.13	61.14	62.64	47.88	59.83	57.18	52.43	57.63
DA5040	54.66	53.81	51.06	51.09	48.02	48.37	48.16	48.66	49.85	63.94	50.62	45.70	50.15	51.63
DY6742	61.76	61.53	61.87	63.11	62.32	62.35	60.70	57.52	58.88	50.66	58.77	60.50	57.22	57.97
DH2600	61.20	60.89	61.53	62.02	63.32	62.85	62.19	61.83	61.32	42.15	56.21	51.68	44.37	52.69
DH5880	42.75	46.54	44.38	42.70	44.41	42.37	41.27	41.36	45.45	40.65	41.70	42.06	26.08	42.73
DT5220	39.27	39.25	36.39	35.83	36.58	36.15	35.18	33.38	36.64	34.36	34.34	33.93	35.00	41.77
DI2440	84.40	83.99	84.72	84.69	84.52	85.15	85.10	84.29	83.79	74.42	83.28	83.78	81.25	75.60
DS6241	57.52	56.52	56.41	58.94	58.30	56.25	60.50	58.20	58.91	36.81	63.45	56.52	56.58	50.76
DM6160	68.08	69.05	69.32	68.99	69.29	70.20	68.71	69.08	68.29	45.48	67.75	67.50	63.87	61.46
DT5360	28.39	30.38	25.81	27.15	31.32	25.37	24.80	25.20	27.32	18.20	25.90	30.36	17.68	32.22
DA0380	79.02	79.86	79.80	80.36	80.78	80.77	80.96	80.42	80.67	82.99	79.24	78.32	77.65	72.73
DW4980	43.87	44.14	42.28	40.52	43.65	40.73	37.62	37.81	40.20	40.63	41.74	41.18	38.89	44.70
DR2700	67.31	67.05	67.14	67.12	66.00	66.65	66.46	66.78	66.94	58.05	67.94	66.89	62.20	63.68
DH5840	40.58	41.49	44.57	41.77	39.98	43.15	40.99	41.44	45.51	41.54	44.52	40.53	27.61	41.19
DA4500	78.69	78.75	78.98	79.39	79.52	79.75	79.95	79.97	80.88	76.70	79.20	78.80	76.73	70.42
DQ9991	75.75	75.99	75.97	74.85	74.54	75.13	75.77	75.45	74.71	59.99	76.31	76.51	75.67	67.96
DP1140	74.46	82.48	81.81	81.14	81.20	76.88	79.53	80.28	79.57	56.48	81.58	82.01	75.81	68.74
DV0860	70.02	69.97	69.10	69.41	69.50	67.87	68.25	67.19	67.38	61.44	67.50	65.40	62.89	66.17
DG1500	81.01	81.88	82.41	82.70	82.64	82.32	82.28	82.25	82.56	77.93	80.66	79.62	82.18	72.54
DG2681	75.63	76.76	76.55	76.58	76.89	74.54	74.51	75.11	74.98	71.86	73.52	72.61	74.51	73.64
DP1662	63.12	63.39	63.87	65.22	66.24	64.41	64.04	64.95	64.77	46.88	66.32	67.28	64.26	58.11
DF5920	33.09	36.06	34.56	30.17	30.57	33.09	33.15	31.54	31.48	39.27	29.53	27.57	24.63	41.20
DN5000	49.98	50.78	51.20	50.74	50.33	50.61	50.59	51.05	50.50	40.43	50.21	48.95	41.95	49.69
DG4501	74.35	73.66	73.39	72.74	72.32	72.67	71.90	72.51	73.38	74.11	74.96	72.71	72.78	69.54
DC4840	16.32	15.84	17.01	11.47	12.69	14.27	8.01	7.23	9.76	31.35	13.14	5.49	6.12	21.46
DC5160	20.80	18.69	16.53	13.49	14.93	15.03	14.40	14.29	18.30	27.76	16.17	10.88	14.12	22.51
DD1180	68.45	67.36	70.38	68.72	70.34	70.65	70.52	69.96	70.32	73.74	68.93	68.51	62.60	64.00

	1400	1300	1200	1100	1000	900	800	700	600	500	400	300	200	100
DA2520	83.63	83.83	84.61	85.03	85.24	84.96	85.12	84.95	85.50	85.75	84.35	82.37	82.53	74.78
DN6060	84.23	83.11	83.05	82.60	82.42	83.02	82.00	81.44	81.15	69.47	81.62	81.72	80.63	73.89
DF5970	35.76	37.06	34.10	35.42	37.93	36.12	37.59	37.68	42.41	49.67	38.19	36.34	31.00	35.33
DD1140	76.62	76.94	77.51	77.32	77.33	77.31	77.94	77.19	76.64	80.24	75.15	75.46	71.23	68.98
DF7740	82.75	82.10	81.90	81.25	81.58	81.49	81.29	80.38	80.33	81.95	81.02	78.80	81.16	74.13
DW5040	41.02	40.28	40.92	40.26	38.95	39.50	37.08	36.72	37.82	36.57	41.66	37.27	40.50	41.86
DF4700	35.94	36.26	35.57	34.07	35.78	33.65	34.57	31.34	33.00	38.81	33.38	33.15	26.86	34.64
DP7560	69.64	70.50	70.36	70.49	70.38	69.67	70.31	70.72	70.17	56.34	67.16	63.60	59.27	62.32
DW3960	55.01	53.85	53.48	52.54	51.25	51.40	49.95	49.78	48.89	53.68	50.92	47.67	48.55	53.48
DA2540	86.14	86.16	86.96	87.31	87.40	87.19	87.19	87.23	87.62	86.96	86.21	84.68	83.91	77.39
DA1540	78.51	79.83	80.73	81.20	81.17	80.35	80.29	79.94	80.48	83.65	78.42	77.72	78.59	71.36
DQ4440	68.08	67.61	69.16	69.81	68.71	68.53	68.44	69.35	68.00	55.41	62.79	61.48	59.57	61.87
DL5560	43.60	43.42	41.27	40.91	44.18	37.37	36.82	35.37	37.12	28.37	35.79	37.90	25.43	41.38
DL2180	60.16	59.55	59.58	54.61	60.90	61.96	59.69	54.91	61.87	42.77	62.14	58.33	62.65	53.73
DX0980	78.22	78.09	77.24	77.70	78.93	77.30	76.65	77.23	75.80	53.41	75.85	74.86	70.63	70.37
DY4480	59.18	58.94	57.36	55.81	59.34	61.52	57.07	52.98	60.12	34.16	51.90	50.75	52.62	53.51
DN1541	84.26	84.58	84.89	84.61	84.46	84.52	84.10	83.93	83.15	69.01	80.63	80.84	81.74	74.31
DR2602	64.86	63.36	64.36	63.57	63.36	63.74	63.44	63.06	63.47	54.79	64.05	63.23	60.06	63.13
DX1660	68.82	68.60	68.84	69.35	69.54	68.78	68.86	70.30	68.66	46.77	69.29	69.83	62.80	59.36
DS2602	53.37	50.75	52.18	48.96	55.83	51.76	48.07	45.85	55.33	34.51	49.57	51.12	47.57	50.69
DV2601	74.84	74.64	75.02	75.57	75.02	74.55	73.83	74.30	74.84	64.55	75.35	75.67	71.69	66.64
DQ2920	63.49	63.31	64.60	65.71	65.34	65.23	65.49	65.82	64.21	53.21	64.66	64.60	64.64	57.90
DS4362	44.28	50.44	47.70	46.85	45.15	45.97	41.97	36.39	42.14	24.74	46.18	41.20	37.87	43.26
DS1520	58.75	59.59	55.74	55.21	56.48	55.19	49.50	50.46	62.23	32.47	53.07	52.97	58.95	50.08
DT3360	71.79	72.68	73.43	73.69	72.81	72.91	72.57	72.86	73.76	64.64	74.45	76.32	72.85	65.06
DG4080	52.83	53.14	53.00	52.81	52.61	52.95	50.74	52.08	52.80	47.30	50.78	51.31	46.48	51.97
DG4700	35.64	38.20	36.38	34.84	34.77	33.71	32.80	33.81	34.76	34.46	36.83	33.21	32.96	36.21
DS6744	52.66	49.43	44.24	50.74	57.17	52.30	52.85	53.46	47.74	23.05	44.08	45.09	46.48	43.91
DN1350	78.25	77.38	76.92	76.73	76.42	76.59	76.62	76.14	75.23	61.40	73.64	74.30	73.33	66.98
DQ6050	79.92	80.58	80.27	80.61	80.05	80.09	80.88	81.17	80.93	73.76	75.82	72.82	67.71	73.01

	1400	1300	1200	1100	1000	900	800	700	600	500	400	300	200	100
DF5400	28.61	32.58	32.09	30.77	31.84	32.79	33.58	33.97	35.01	34.28	32.70	33.77	26.69	35.97
DD2600	71.12	70.98	71.05	70.80	71.21	71.52	71.75	72.12	73.02	68.95	72.50	72.84	72.40	66.99
DS1664	47.34	47.87	46.14	46.25	45.88	47.76	44.30	42.99	46.94	25.05	42.84	38.11	40.23	39.72
DS3880	44.52	49.15	47.74	46.26	46.49	50.17	44.95	38.85	47.90	28.11	45.23	45.29	42.26	44.60
DP5120	43.90	41.65	40.95	40.60	39.23	37.23	36.47	37.37	36.48	30.89	37.96	35.78	36.46	41.26
DR4800	12.41	3.92	8.24	7.09	4.60	5.35	2.44	1.61	3.57	6.66	4.69	3.39	5.57	8.99
DP5200	30.27	30.45	29.10	27.16	28.60	26.33	23.45	23.67	26.48	26.12	25.69	25.78	26.85	33.78
DN5160	53.98	56.53	57.17	53.49	56.67	56.79	57.33	56.79	57.06	39.06	51.71	56.10	46.50	51.30
DE2500	60.67	61.34	61.61	62.05	61.53	61.76	62.87	61.26	61.56	64.43	58.28	54.93	56.07	56.80
DZ7560	24.16	5.65	9.50	7.94	11.37	8.30	7.32	6.89	12.48	4.60	7.46	11.27	7.79	28.12
DG5120	32.01	31.80	28.85	29.88	28.50	29.15	28.66	26.80	28.76	38.88	28.17	31.34	32.55	34.80
DX3720	13.81	4.95	6.71	15.31	8.26	6.03	5.78	8.50	8.16	-6.08	9.58	-3.40	-0.51	21.77
DE4980	35.16	41.65	39.79	37.15	38.25	36.03	37.71	37.71	41.56	49.34	42.05	39.17	37.44	37.00
DW2720	72.48	80.09	79.07	78.35	78.15	74.04	76.18	76.43	76.38	71.36	78.99	79.10	74.70	71.16
DF6280	77.32	76.72	76.07	75.31	75.82	76.16	76.01	75.18	74.71	77.37	75.84	74.15	76.42	70.06
DP8041	65.13	66.16	65.86	65.90	66.16	65.95	66.39	67.24	66.42	48.42	63.55	59.98	54.74	58.03

Note: the first row gives the number of training data to construct a SCFG; the first column gives the test data code (Eddy 1994); the remaining entries contain *the negative log likelihood* (*NLL*) score of the sequence(Eddy 1994); the *NLL* score = -log(Prob($s \mid G$)), the probability of a sequence given by the grammar *G*.

VITA AUCTORIS

Kaiyuan Shi was born in 1974 in Xinwen county, Sichuan province, People's Republic of China. He graduated from North China Institute of Water Conservancy & Hydroelectric Power, China, where he obtained a B.Eg. in Computer Science in 1998.

He is currently a candidate for the Master's degree in Computer Science at the University of Windsor and hopes to graduate in Fall 2003.

He served as the Senator of the Graduate Student Society (GSS) at the University of Windsor from May 2002 to April 2003.

He also served as the President of the Chinese Students and Scholars Association (CSSA) at the University of Windsor from October 2001 to September 2002.

He worked as a software developer for three years in IT area after he obtained his bachelor degree in Computer Science.

Before came to Canada, he worked in Shenzhen Modern Computer Manufacturer (MCM), China.